TUGAS AKHIR – KI141502

Studi Perbandingan Kinerja Model Transmisi *TwoRayGround* dan Nakagami pada Optimized Link State Routing (OLSR) di Lingkungan Mobile Ad Hoc Network (MANET) menggunakan Network Simulator 2 (NS-2)

DHIYA'AN SABILA RAMADHANI
NRP 5110 100 017

Dosen Pembimbing
Dr. Eng. RADITYO ANGGORO, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

TUGAS AKHIR – KI141502

Studi Perbandingan Kinerja Model Transmisi *TwoRayGround* dan Nakagami pada Optimized Link State Routing (OLSR) di Lingkungan Mobile Ad Hoc Network (MANET) menggunakan Network Simulator 2 (NS-2)

DHIYA'AN SABILA RAMADHANI
NRP 5110 100 017

Dosen Pembimbing
Dr. Eng. RADITYO ANGGORO, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

Comparative Study of *TwoRayGround* and Nakagami Propagation Model for Optimized Link State Routing (OLSR) on Mobile Ad Hoc Network (MANET) using Network Simulator 2 (NS-2)

DHIYA'AN SABILA RAMADHANI
NRP 5110 100 017

Advisor
Dr. Eng. RADITYO ANGGORO, S.Kom., M.Sc.

INFORMATICS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

STUDI PERBANDINGAN KINERJA MODEL TRANSMISI *TWORAYGROUND* DAN NAKAGAMI PADA OPTIMIZED LINK STATE (OLSR) DI LINGKUNGAN MOBILE AD HOC NETWORK (MANET) MENGGUNAKAN NETWORK SIMULATOR 2 (NS-2)

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Arsitektur Jaringan Komputer
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember Surabaya

Oleh:
DHIYA'AN SABILA RAMADHANI
NRP. 5110 100 017

Disetujui oleh Pembimbing Tugas Akhir:

Dr. Eng. Radityo Anggoro, S.Kom., M.Sc. .....
NIP: 198410162008121002 (pembimbing)

SURABAYA
JUNI, 2016

[Halaman ini sengaja dikosongkan]

STUDI PERBANDINGAN KINERJA MODEL TRANSMISI *TWORAYGROUND* DAN NAKAGAMI PADA OPTIMIZED LINK STATE (OLSR) DI LINGKUNGAN MOBILE AD HOC NETWORK (MANET) MENGGUNAKAN NETWORK SIMULATOR 2 (NS-2)

Nama Mahasiswa	**: Dhiya'an Sabila Ramadhani**
NRP	**: 5110 100 017**
Jurusan	**: Teknik Informatika FTIf - ITS**
Dosen Pembimbing	**: Dr. Eng. Radityo Anggoro, S.Kom., M.Sc.**

ABSTRAK

Perangkat mobile seperti notebook, handphone, tablet dan lain lain mulai berkembang adanya teknologi nirkabel (wireless) saat ini yang menjadi indikator kemajuan peradaban manusia memungkinkan perangkat komunikasi dapat berkomunikasi secara langsung dengan perangkat lainnya dalam posisi bergerak dan tanpa adanya jaringan infrastruktur yang tetap dan bersifat sementara, jaringan semacam ini disebut sebagai MANET (Mobile Ad Hoc Network).

Dalam MANET, setiap node bergerak secara bebas, sehingga jaringan dapat mengalami perubahan topologi dengan cepat. Karena node dalam MANET memiliki jarak transmisi yang terbatas, beberapa node tidak bisa berkomunikasi secara langsung dengan node lainnya. Namun implementasi MANET di dunia nyata masih sulit untuk dilakukan sehingga banyak penelitian dilakukan dengan membuat simulasi MANET menggunakan network simulator.

Maka dari itu, pada Tugas Akhir ini yang diteliti adalah skema MANET yang dihasilkan oleh file node-movement dan traffic-pattern yang telah ada pada distribusi network simulator. Penelitian ini menggunakan NS-2 sebagai network simulator dengan protokol proaktif MANET jenis OLSR (Optimized Link

State Protocol) *sebagai protokol routing yang digunakan serta menggunakan model transmisi TwoRayGround dan Nakagami sebagai pembanding yang ada pada NS-2.*

Kemudian sistem pada Tugas Akhir ini diuji fungsionalitas dan performanya dengan melalui beberapa skenario yang telah ditentukan. Hasil dari pengujian adalah suatu perbandingan model transmisi TwoRayGround dan Nakagami pada protokol routing OLSR di lingkungan MANET dengan menggunakan NS-2. Performa protokol routing tersebut diukur berdasarkan routing overhead, packet delivery ratio, dan delay pengiriman paket dari satu node ke node lainnya.

Kata Kunci: MANET, Nakagami, Network Simulator, NS-2, OLSR, TwoRayGround.

COMPARATIVE STUDY OF *TWORAYGROUND* AND NAKAGAMI PROPAGATION MODEL FOR OPTIMIZED LINK STATE *ROUTING* (OLSR) ON MOBILE AD HOC NETWORK USING NETWORK SIMULATOR 2 (NS-2)

Name	**: Dhiya'an Sabila Ramadhani**
NRP	**: 5110 100 017**
Major	**: Informatics Engineering, IT Dept – ITS**
Advisor	**: Dr. Eng. Radityo Anggoro, S.Kom., M.Sc.**

ABSTRACT

Mobile devices such as notebooks, mobile phones, tablets and others began to grow in the presence of wireless technologies which is now becoming the indicators of human civilization advancement that allows communication devices to communicate directly with other devices in positional moves and without any fixed or temporary network infrastructure, this kind of networks are called as MANET (Mobile Ad Hoc Network).

On the MANET, each node moves random, so the network topology may change rapidly. Because the nodes in MANET has a limited transmission distance, some nodes can not communicate directly with other nodes. But implementation of MANET in the real world, is still difficult to do, so many research was done by making a MANET simulation using network simulator.

Therefore, in this final project studied the scheme of MANET which generated by the file node-movement and traffic pattern that already available in the distribution of network simulator. This study uses the NS-2 as a network simulator with proactive MANET protocol type OLSR (Optimized Link State Protocol) as the routing protocol and transmission models of TwoRayGround and Nakagami as a comparison that exist in NS-2.

Then the system in this final project tested the functionality and performance through a few scenarios that have been determined. The results of the test is a transmission model comparison between TwoRayGround and Nakagami on OLSR routing protocol in MANET environments by using NS-2. The routing protocol performance is measured based on routing overhead, packet delivery ratio, and delay delivery of packets from one node to another.

Keywords: MANET, Nakagami, Network Simulator, NS-2, OLSR, TwoRayGround.

KATA PENGANTAR

Bismillahirrohmanirohim.

Alhamdulilahirabil'alamin, segala puji bagi Allah SWT, atas segala rahmat dan karunia-Nya yang tak terhingga sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"STUDI PERBANDINGAN KINERJA MODEL TRANSMISI *TWORAYGROUND* DAN NAKAGAMI PADA OPTIMIZED LINK STATE (OLSR) DI LINGKUNGAN MOBILE AD HOC NETWORK (MANET) MENGGUNAKAN NETWORK SIMULATOR 2 (NS-2)".

Terselesaikannya Tugas Akhir ini tidak terlepas dari bantuan banyak pihak. Oleh karena itu melalui lembar ini penulis ingin mengucapkan terima kasih dan penghormatan sebesar-besarnya kepada pihak-pihak sebagai berikut.

1. Allah SWT, karena limpahan rahmat dan karunia-Nya lah penulis dapat menyelesaikan Tugas Akhir dan juga perkuliahan di Teknik Informatika ITS.
2. Ummi penulis, Dewi Kurniasari yang tiada henti memberikan semangat, doa serta dukungan penuh kepada penulis selama ini sehingga dapat menyelesaikan Tugas Akhir ini dan perkuliahan di Teknik Informatika ITS Surabaya.
3. Adik penulis, Luthfiyyah Afanin dan Aisyah Humairo yang telah memberikan dukungan dan doa kepada penulis untuk terus semangat dalam penyelesaian Tugas Akhir ini.
4. Bapak Dr. Eng. Radityo Anggoro S.Kom., M.Sc. selaku dosen pembimbing dari penulis yang telah memberikan bimbingan, dukungan, masukan, nasihat dan banyak

arahan kepada penulis dalam menyelesaikan Tugas Akhir ini.

5. Bapak Daniel O. Siahaan, S.Kom., M.Sc., PDEng. selaku dosen wali dari penulis yang selalu memberi nasihat kepada penulis selama menjalani perkuliahan di Teknik Informatika ITS.
6. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku Ketua Jurusan Teknik Informatika ITS.
7. Sahabat tersayang dari penulis, Kessya Din Dalmi yang selalu mendengarkan keluh kesan penulis, memberikan dukungan, hiburan, semangat, nasihat dan menemani keseharian penulis serta mengingatkan penulis agar cepat menyelesaikan Tugas Akhir.
8. Andrianto, yang selalu setia menemani, membantu, memberikan dukungan, doa serta semangat yang tiada henti kepada penulis dengan penuh sabar, suka, duka serta kepedulian yang sangat besar kepada penulis.
9. Sahabat super sepermainan dari penulis, Wildhan, Fitri, Haryo, Aji, Caca, Dimas, Wido, Dicky, dan Ruka yang selalu menguatkan, memberikan dukungan, semangat, nasihat, hiburan dan canda tawa kepada penulis hingga saat ini.
10. Verina Ervan Kardjani dan Rahmadhianty, tante terfavorit dan adik sepupu dari penulis yang selalu memberikan motivasi, dukungan dan semangat selama mengerjakan Tugas Akhir serta menemani keseharian penulis.
11. Bima Bahteradi, teman seperjuangan dalam mengerjakan Tugas Akhir yang selalu bimbingan bersama.
12. Nyoman Bagus Pratistha, yang mau sudah mau direpotkan dengan pertanyaan-pertanyaan yang berhubungan dengan topik Tugas Akhir dari penulis.
13. Teman-teman TC angkatan 2010, kakak angkatan, dan juga adik angkatan yang telah ramah dan berbaik hati membantu penulis selama berada di Teknik Informatika.

Terima kasih atas rasa kekeluargaan yang telah kalian berikan kepada penulis.

14. Pihak-pihak yang tidak dapat penulis sebutkan satu per satu yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa laporan Tugas Akhir ini masih jauh dari kata sempurna. Oleh karena itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan penulis ke depannya. Penulis berharap laporan Tugas Akhir ini dapat berguna bagi pembaca secara umum. Semoga Allah SWT. memberkati dan membalas semua kebaikan yang telah dilakukan

Surabaya, Juni 2016

Dhiya'an Sabila Ramadhani

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Bab ini memaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Seiring dengan perkembangan zaman, teknologi informasi berkembang dengan pesatnya dan kebutuhan masyarakat akan komunikasi dan mengakses informasi pun semakin mudah. Perangkat *mobile* seperti *notebook, handphone, tablet* dan lain lain mulai berkembang adanya teknologi nirkabel (*wireless*) saat ini yang menjadi indikator kemajuan peradaban manusia memungkinkan perangkat komunikasi dapat berkomunikasi secara langsung dengan perangkat lainnya dalam posisi bergerak dan tanpa adanya jaringan infrastruktur yang tetap. Teknologi jaringan nirkabel (*wireless*) dapat membuat beberapa perangkat *mobile* tersebut yang berada di dalam suatu area dimana fasilitas nirkabel tersebut bisa saling terkoneksi dan saling menjangkau akan sangat mungkin membentuk sebuah jaringan yang bersifat sementara dan jaringan semacam ini disebut sebagai *Mobile Ad Hoc Network* (MANET).

Jaringan *Mobile Ad Hoc Network* (MANET) merupakan suatu jaringan yang terorganisir secara mandiri tanpa adanya dukungan infrastruktur. Dalam MANET, setiap *node* bergerak secara bebas, sehingga jaringan dapat mengalami perubahan topologi dengan cepat. Karena *node* dalam MANET memiliki jarak transmisi yang terbatas, beberapa *node* tidak bisa berkomunikasi secara langsung dengan *node* lainnya. Pada MANET, jalur *routing* mengandung beberapa *hop* dan setiap *node* berfungsi sebagai *router* untuk menentukan ke arah mana tujuan atau rute yang akan mereka pilih. Dalam menentukan setiap jalur *routing* pada MANET terdapat tiga jenis protokol *routing* yang diklasifikasikan menjadi tiga, diantaranya protokol

routing proactive, reactive dan *hybrid*. Pada protokol *routing proactive* akan terus mempelajari perubahan topologi secara *real-time* melalui *node* jaringan tetangganya. Oleh karena itu, setiap ada permintaan rute dari *node* sumber ke *node* tujuan, informasi *routing* tersebut sudah tersedia pada *node* sumber. Seiring dengan perubahan topologi jaringan, akan dapat terjadinya peningkatan keseluruhan biaya pemeliharaan jaringan [1].

Implementasi pada lingkungan MANET dapat dilakukan dengan menggunakan simulasi sehingga penelitian ini dapat dilakukan untuk mempelajari sistem dengan baik. Simulasi dilakukan dengan menggunakan Network Simulator 2 (NS-2). Implementasi ini akan dilakukan analisa performa protokol *routing proactive* yaitu *Optimized Link State Routing* (OLSR). Pada kasus ini menggunakan model transmisi *TwoRayGround* dan Nakagami digunakan sebagai pembanding yang ada pada NS-2.

Hasil yang diharapkan dari Tugas Akhir ini adalah suatu perbandingan model transmisi *TwoRayGround* dan Nakagami pada protokol *routing* OLSR di lingkungan MANET dengan menggunakan aplikasi. Performa protokol *routing* tersebut diukur berdasarkan *routing overhead, packet delivery ratio,* dan *delay* pengiriman paket dari *node* ke *node* lainnya.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut.

1. Bagaimana kinerja protokol *routing* OLSR pada MANET?
2. Bagaimana perbandingan model transmisi *TwoRayGround* dan Nakagami pada OLSR di lingkungan MANET?

1.3. Batasan Permasalahan

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, di antaranya sebagai berikut.

1. Protokol *routing* hanya dijalankan dan diujicobakan pada aplikasi *Network Simulator 2* (NS-2)
2. Protokol *routing* yang diujicobakan adalah OLSR.
3. Lingkungan jaringan digunakan untuk uji coba adalah *Mobile Ad Hoc Network* (MANET).
4. Model transmisi yang akan dibandingkan dalam Tugas Akhir ini adalah *TwoRayGround* dan Nakagami.

1.4. Tujuan dan Manfaat

Tujuan dari pembuatan Tugas Akhir ini adalah untuk memberikan hasil perbandingan studi kerja model transmisi *TwoRayGround* dan Nakagami pada protokol *routing* OLSR di lingkungan MANET dengan menggunakan aplikasi *Network Simulator* 2 (NS-2).

Dengan dibuatnya Tugas Akhir ini akan memberikan sebuah manfaat dalam menentukan model transmisi yang tepat untuk pengiriman data pada lingkungan MANET dengan implementasi protokol *routing* OLSR.

1.5. Metodologi

Tugas Akhir ini menggunakan beberapa tahapan dalam proses pengerjaannya. Metodologi yang dilakukan dalam pengerjaan Tugas Akhir ini terdiri atas beberapa tahapan yang dipaparkan sebagai berikut.

1. **Penyusunan Proposal Tugas Akhir**
   Pada tahap ini dilakukan penyusunan proposal Tugas Akhir yang berisi tentang deskripsi umum rancangan Tugas Akhir yang akan dibuat. Penyusunan ini terdiri dari menentukan judul Tugas Akhir, latar belakang, rumusan masalah, batasan masalah, tujuan Tugas Akhir, manfaat Tugas Akhir, tinjauan pustaka, ringkasan Tugas Akhir, metodologi, serta rencana jadwal kegiatan pengerjaan Tugas Akhir.

2. **Studi Literatur**
   Pada tahap ini dilakukan studi literatur mengenai *tools* dan metode yang digunakan. Literatur yang dipelajari dan

digunakan meliputi buku referensi, artikel, jurnal dan dokumentasi dari internet.

3. **Implementasi Protokol *Routing***
   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini merupakan tahap yang paling penting dimana bentuk awal aplikasi yang akan diimplementasikan didefinisikan. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada.

4. **Pengujian dan Evaluasi**
   Pada tahap ini dilakukan uji coba terhadap aplikasi yang telah dibuat. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuain dengan perencanaan. Tahap ini dimaksudkan juga untuk mengevaluasi jalannya sistem, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

5. **Penyusunan Buku Tugas Akhir**
   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam Tugas Akhir ini serta hasil dari implementasi aplikasi yang telah dibuat. Sistematika penulisan buku Tugas Akhir secara garis besar antara lain sebagai berikut.

   1. Pendahuluan
      a. Latar Belakang
      b. Rumusan Permasalahan
      c. Batasan Permasalahan
      d. Tujuan dan Manfaat
      e. Metodologi
      f. Sistematika Penulisan
   2. Tujuan Pustaka
   3. Perancangan Sistem
   4. Implementasi

5. Pengujian dan Evaluasi
6. Penutup

1.6. Sistematika Penulisan

Buku Tugas Akhir ini terdiri atas beberapa bab yang dijelaskan sebagai berikut.

- Bab I. Pendahuluan
  Bab ini berisi latar belakang masalah, permasalahan, batasan masalah, tujuan Tugas Akhir, manfaat Tugas Akhir, metodologi yang digunakan, dan sistematika penyusunan buku Tugas Akhir.
- Bab II. Tinjauan Pustaka
  Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.
- Bab III. Perancangan
  Bab ini berisi perancangan metode yang nantinya akan diimplementasikan dan dilakukan pengujian dari aplikasi yang akan dibangun.
- Bab IV. Implementasi
  Bab ini membahas implementasi dari rancangan sistem atau desain yang dilakukan pada tahap perancangan. Penjelasan berupa implementasi skenario mobilitas *node-node* pada jaringan *wireless* yang tidak mempunyai *router* tetap yang dibuat menggunakan *file node-movement* dan *traffic-pattern* yang ada pada *network simulator*, konfigurasi sistem dan skrip analisis yang digunakan untuk menguji performa protokol *routing*.

- Bab V. Pengujian dan Evaluasi
  Bab ini menjelaskan tahap pengujian sistem dan performa dalam skenario mobilitas *ad hoc* yang dibuat oleh distribusi *mobility* dalam *network simulator*.
- Bab VI. Penutup
  Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan terhadap rumusan masalah yang ada serta saran untuk pengembangan selanjutnya.

BAB II
TINJAUAN PUSTAKA

Bab ini membahas mengenai dasar-dasar teori yang berkaitan dengan pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran atau definisi secara umum terhadap alat, protokol *routing* serta definisi yang digunakan dalam pembuatan Tugas Akhir.

2.1. *Mobile Ad Hoc Network* (MANET)

Mobile Ad Hoc Network (MANET) adalah kumpulan dari beberapa *wireless node* yang dapat di *set-up* secara dinamis dimana saja dan kapan saja, tanpa menggunakan infrastruktur jaringan yang ada. MANET juga merupakan jaringan sementara yang dibentuk oleh beberapa *mobile node* tanpa adanya pusat administrasi dan infrastruktur kabel. Pada MANET, *mobile host* yang terhubung dengan *wireless* dapat bergerak bebas dan juga berperan sebagai *router*. MANET memiliki beberapa karakteristik yaitu di antaranya konfigurasi jaringan yang dinamis, *bandwidth* yang terbatas, keterbatasan daya untuk tiap-tiap operasi, keterbatasan keamanan dan setiap *node* pada MANET berperan sebagai *end-user* sekaligus sebagai *router* yang menghitung sendiri *route-path* yang selanjutnya akan dipilih [2].

Di zaman smartphone sekarang ini, MANET bisa jadi hal yang sangat membantu proses pertukaran data antar *node*. Misal, jika kita ingin melakukan komunikasi data dengan 10 orang yang berada di daerah terbuka, jauh dari jaringan internet, maka MANET menjadi solusi terbaik. Sebab kita tidak harus mengandalkan jaringan internet sebagai pusat penghubungnya. Setiap *mobile node* akan dapat berkoneksi secara langsung [3].

Pada Gambar 2.1 Skema MANET ditunjukkan pada penerapan *Ad-Hoc* yang dibuat dari perangkat *mobile device*. Untuk arsitekur pada MANET sendiri pada lapis protokol MANET yang punya kemiripan dengan layer TCP/IP tampak ada perbedaan di layer *network*-nya. *Mobil*e *node* menggunakan *ad-hoc* protokol *routing* untuk merutekan paket-paket. Layer network

MANET dibagi menjadi dua bagian yaitu layer *network* dan layer *ad-hoc routing*. Protokol yang digunakan pada bagian network layer MANET adalah Internet Protokol (IP) dan protokol yang dipakai pada bagian *ad-hoc routing* layer *adalah Ad Hoc On-Demand Distance Vector* (AODV). Protokol *routing* ad hoc yang lainpun bisa juga dipakai pada layer ini. Pada layer *transport* digunakan *User Datagram Protokol* (UDP). Karakteristik yang dimiliki UDP diantaranya toleran terhadap *loss* dan tidak toleran terhadap *delay*. Pengiriman paket ini tidak memerlukan paket balasan yang berfungsi untuk memastikan sampainya sebuah paket. Meskipun demikian paket-paket UDP mempunyai keunggulan yaitu lebih efektif dalam penggunaan *bandwidth*, karena mampu meneruskan paket ke jalur lain apabila terjadi suatu kemacetan dalam pengiriman paket [5].

Gambar 2.1 Skema MANET [4]

Terdapat berbagai jenis protokol *routing* untuk MANET yang secara keseluruhan dapat dibagi menjadi beberapa kelompok, antara lain:

a. *Proactive Routing*
   Algoritma ini akan mengelola daftar tujuan dan rute terbaru masing-masing dengan cara mendistribusikan *routing table* ke seluruh jaringan, sehingga jalur lalu lintas (*traffic*) akan sering dilalui oleh *routing table* tersebut. Hal ini akan memperlambat aliran data jika terjadi restrukturisasi *routing table*. Contoh *proactive routing* adalah Babel, *Intrazone Routing Protocol* (IARP), dan *Optimized Link State Routing* (OLSR).
b. *Reactive Routing*
   Tipe ini akan mencari rute (*on demand*) dengan cara membanjiri jaringan dengan paket *router request.* Sehingga dapat menyebabkan jaringan akan penuh (*clogging*). Contoh *reactive routing* adalah *Ad Hoc On Demand Distance Vector* (AODV), *Dynamic MANET On Demand Routing* (DYMO).
c. *Flow Oriented Routing*
   Tipe protokol ini mencari rute dengan mengikuti aliran yang disediakan. Salah satu pilihan adalah dengan unicast secara terus-menerus ketika meneruskan data saat mempromosikan link baru. Beberapa kekurangan tipe protokol ini adalah membutuhkan waktu yang lama untuk mencari rute yang baru. Contoh *flow oriented routing* adalah *Interzone Routing Protocol* (IERP), *Lightweight Underlay Network Ad Hoc Routing* (LUNAR), *Signal Stability Routing* (SSR).
d. *Hybrid Routing*
   Tipe protokol ini menggabungkan antara *proactive routing* dengan *reactive routing.* Contoh *hybrid routing* adalah *Hybrid Routing Protocol for Large Scale MANET* (HRPLS), *Hybrid Wireless Mesh Protocol* (HWMP), *Zone Routing Protocol* (ZRP) [2].

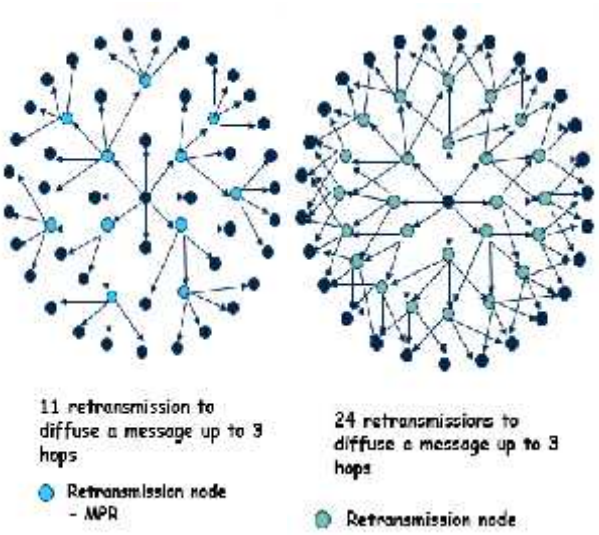
2.2. *Optimized Link State Routing* (OLSR)

OLSR merupakan protokol *routing* proaktif, yang dapat dengan segera menyediakan *routing* ke semua *network* tujuan yang ada. Protokol ini merupakan pengembangan dari algoritma *link-state* klasik untuk memenuhi persyaratan dari jaringan nirkabel dinamis seperti MANET. OLSR merupakan protokol *routing* yang menggunakan algoritma *link-state* klasik dan algoritma Djikstra untuk mencari *shortest path*. Optimalisasi ini berdasarkan pada konsep *Multipoint Relays* (MPR). Setiap *node* menyeleksi *node-node* tetangganya sebagai *Multipoint Relay* (MPR). Pada OLSR, hanya *node* yang berperan sebagai MPR yang bertanggung jawab untuk melanjutkan *control traffic* (paket kontrol), yang dimaksud untuk penyebaran ke seluruh jaringan. *Multipoint Relays* (MPRs) menyediakan mekanisme untuk melakukan *flood control traffic* dengan mengurangi jumlah transmisi yang dibutuhkan.

Pertama dengan menggunakan *multipoint relay* dapat mengurangi ukuran dari *control message*. Daripada menyatakan semua *link*, *node* menyatakan hanya sekumpulan *links* dengan *node* tetangganya sebagai *"multipoint relay"*. Penggunaan MPR juga meminimalisasi *flooding* dari *control traffic*. Teknik ini secara signifikan mengurangi jumlah transmisi ulang dari *broadcast control* message. Sistem link state mempunyai cara yang lebih efisien dibanding dengan sistem *distance vector*. Jika ada proses pembaharuan, maka hanya pembaharuan itu yang akan dikirimkan. Koleksi jalur terbaik kemudian akan membentuk tabel *routing node*.

Sebuah *node* OLSR yang sedang beroperasi akan secara periodik menyebarkan paket "*hello*" sehingga tetangga dapat mendeteksi keberadaan *node* tersebut. Setiap *node* menghitung berapa jumlah paket "*hello*" yang hilang atau diterima dari tetangga sehingga mendapatkan informasi tentang topologi dan kualitas sambungan *node* lingkungan. Informasi topologi yang diterima di *broadcast* sebagai pesan *topology control* (TC) dan

diteruskan oleh tetangga yang dipilih sebagai *multipoint-relay* (MPR).



Gambar 2.2 Proses *Packet Flooding* pada OLSR [6]

Pada Gambar 2.2 melalui MPR, hanya *node* yang dipilih saja yang bertugas untuk membanjiri jaringan dengan mengirimkan *broadcast-messages* ke seluruh jaringan.

OLSR tepat digunakan untuk jaringan yang besar dan padat karena penggunaan MPRs. Semakin besar dan padat suatu jaringan, maka optimasi akan semakin berpengaruh dibanding dengan protokol *link-state* klasik. OLSR menggunakan proses pencarian rute *hop-by-hop* dimana setiap *node* menggunakan informasi *routing table* lokalnya untuk mengirim paket. OLSR lebih efisien apabila bekerja dalam jaringan padat dengan *traffic* yang rendah. OLSR membutuhkan *bandwith* yang konstan untuk memperoleh *topology update message* [6]. Protokol OLSR digunakan sebagai protokol MANET.

2.3. Model Transmisi *TwoRayGround*

Dalam *mobile radio channel*, *single direct path* antara *base station* dan *mobile* terkadang hanya peralatan fisik biasa untuk propagasi dan rumus pada *free space* kurang akurat jika dalam penggunaannya berdiri sendiri. Model transmisi *TwoRayGround* merupakan model yang berguna karena berdasar pada optik geometri dan dapat digunakan untuk *direct path* dan refleksi dari *ground* antara *transmitter* dan *receiver*. Model ini dirasa sangat akurat untuk memperkirakan kekuatan sinyal dalam skala luas dengan jarak beberapa kilometer untuk sistem *mobile radio* dengan menggunakan menara yang tinggi. *Power* yang diterima dengan jarak d diberikan oleh persamaan (1):

$$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L} \quad \textbf{(1)}$$

dimana P_t adalah power yang ditransmisikan, G_t dan G_r adalah tegangan antena pada *transmitter* dan *receiver*, h_t dan h_r adalah tinggi dari antena *transmitter* dan *receiver*, nilai L diasumsikan sama dengan nilai L pada propagasi *free space*, L = 1. Untuk parameter yang lain, masih sama dengan parameter pada propagasi *free space*. Berdasarakan rumus matematika diatas, *power loss* lebih cepat hilang dibandingkan dengan rumus matematika pada model propagasi *free space* ketika jaraknya bertambah. Namun, Model ini tidak memberikan hasil yang baik untuk jarak yang terlalu dekat dikarenakan osilasi yang disebabkan oleh konstruktif dan destruktif yang merupakan kombinasi dari model ini [7].

2.4. Model Transmisi Nakagami

Model Nakagami-m-m bersifat lebih umum dan dapat diterapkan untuk berbagai kondisi fading, tergantung pada parameter-m yang digunakan. Kanal Nakagami-m-m memiliki *probability density function* (PDF) yang dinyatakan sebagai persamaan (2) [8].

$$P_R(R) = \frac{2m^m R^{2m-1}}{\Gamma(m)\Omega^m} e^{(-\frac{m}{\Omega})}, \qquad R \geq 0 \tag{2}$$

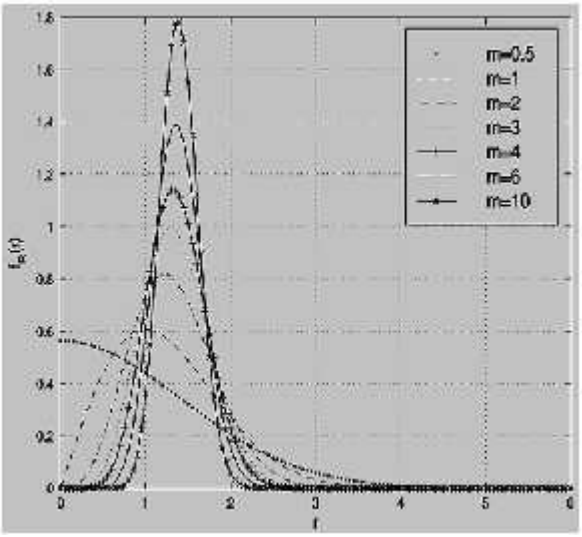
dimana :
m = parameter fading, m ≥ 0.5 sampai ~ (integer positif)
R = amplitudo fading
Γ = fungsi gamma
Ω = E [R^2]

Pengaruh parameter-m pada kanal Nakagami-m-m :

1. Apabila $m \leq 1$, maka persamaan (3) menjadi *probabilitas density function* dari kanal fading Rayleigh.
2. Apabila $m>1$, merujuk ke kanal fading Ricean. Dimana kanal fading Rician memiliki faktor K yang memiliki pengaruh disini. Sehingga nilai m adalah sebagai persamaan (3) [8].

$$m = \frac{(1+k)^2}{1+2k}, \qquad k \geq 0 \tag{3}$$

Pengaruh dari pada parameter-m dapat dilihat pada grafik *Probability Density Function* (PDF) kanal Nakagami-m-m seperti pada Gambar 2.3.



Gambar 2.3 PDF Kanal Nakagami-m-m untuk Parameter-*m* yang Berbeda [8]

2.5. *Network Simulator 2* (NS-2)

Network Simulator 2 (NS-2) merupakan alat simulasi jaringan yang bersifat *open source* yang banyak digunakan dalam mempelajari struktur dinamik dari jaringan komunikasi. Simulator ini ditargetkan pada penelitian jaringan dan memberikan dukungan yang baik untuk simulasi *routing*, protokol *multicast* dan protokol IP, seperti UDP, TCP, RTP, jaringan nirkabel dan jaringan satelit. Beberapa keuntungan menggunakan *network simulator* sebagai perangkat lunak simulasi yaitu *network simulator* dilengkapi dengan *tools* validasi, pembuatan simulasi dengan menggunakan *network simulator* jauh lebih mudah daripada menggunakan *software developer* seperti Delphi atau C++, *network simulator* bersifat *open source* di bawah GPL (Gnu Public License) dan dapat digunakan pada sistem operasi Windows dan sistem operasi Linux [9].

Pada Tugas Akhir ini digunakan NS-2 versi 2.35 sebagai aplikasi simulasi jaringan skenario MANET yang dihasilkan oleh

program *default* dari NS-2 yaitu *generator file node-movement* dan *traffic-connection pattern* menggunakan protokol OLSR. NS-2 dijalankan pada sistem operasi Linux dan proses instalasinya akan disajikan pada bagian Lampiran.

2.6. *Generator File Node-Movement* dan *Traffic-Connection Pattern*

2.6.1. *File Node-Movement (Mobility Generator)*

Tools yang disebut 'setdest' dikembangkan oleh CMU (Carnegie Mellon University) untuk menghasilkan *random movement* dari *node* dalam jaringan nirkabel. *Node movement* dihasilkan dengan kecepatan gerak yang spesifik menuju lokasi acak atau lokasi spesifik yang berada dalam kawasan yang telah ditentukan. Ketika *node* tiba ke lokasi pergerakan, *node* tersebut bisa diatur untuk berhenti untuk sementara waktu. Setelah itu, *node* terus bergerak menuju lokasi berikutnya. Lokasi 'setdest' berada pada direktori '~ns/indep-utils/cmu-scen-gen/setdest/'

Pengguna harus menjalankan program 'setdest' sebelum menjalankan program simulasi. Format *Command Line* 'setdest' ditunjukkan pada Gambar 2.4 dan keterangannnya ditunjukkan pada Tabel 2.1.

```
./setdest [-v version ] [-n num_of_nodes] [-p
pausetime] [-M maxspeed] [-t simtime] [-x maxx] [-y
maxy] > [outdir/movement-file]
```

Gambar 2.4 Format *Command Line* 'setdest'

Tabel 2.1 Keterangan pada *Command Line* 'setdest'

Parameter	Keterangan
-v version	Versi 'setdest' simulator yang digunakan
-n num	Jumlah *node* dalam skenario
-p pausetime	Durasi ketika sebuah *node* tetap diam setelah tiba di lokasi pergerakan. Jika nilai ini diatur ke 0, maka *node* tidak akan berhenti ketika tiba di lokasi

	pergerakan dan akan terus bergerak
Parameter	**Keterangan**
-M maxspeed	Kecepatan maksimum sebuah *node*. *Node* akan bergerak pada kecepatan acak dalam rentang [0, amxspeed]
-t simtime	Waktu simulasi
-x max x	Panjang maksimum area simulasi
-y max y	Lebar maksimum area simulasi

Command Line 'setdest' menghasilkan *file output* yang berisi jumlah *node* dan mobilitas yang akan digunakan dalam *file* Tcl selama simulasi. *File output*, selain mengandung skrip pergerakan, juga mengandung beberapa statistik lain tentang perubahan *link* dan rute.

Untuk membuat skenario *node-movement* yang terdiri dari 50 *node*, bergerak dengan kecepatan maksimum 20.0 m/s dengan jeda rata-rata antar gerakan sebesar 2 detik, simulasi akan berhenti setelah 200 detik dengan batas topologi yang diartikan sebagai 500 x 500 meter^2, *command line*-nya terlihat seperti seperti pada Gambar 2.5

```
./setdest -v 1 -n 50 -p 2.0 -M 20.0 -t 200 -x 500
-y 500 > scen-20-test
```

Gambar 2.5 Contoh *Command Line* 'setdest'

File output ditulis ke "stdout" secara *default*. Di sini *output* disimpan ke dalam *file* "scen-20-test". *File* dimulai dengan posisi awal *node* dan berlanjut menetapkan *node-movement* seperti terlihat pada Gambar 2.6.

```
$ns_ at 2.000000000000 "$node_(0) setdest
90.441179033457 44.896095544010
1.373556960010
```

Gambar 2.6 Hasil *Output* pada *file* 'scen-20-test'

Command line pada Gambar 2.6 dari 'scen-20-test' mendefinisikan bahwa *node* (0) pada detik ke 2.0 mulai bergerak ke arah tujuan (90.44, 44. 89) dengan kecepatan 1.37 m/s. *Command line* ini dapat digunakan untuk mengubah arah dan kecepatan gerak dari *mobile node*. Arahan untuk *General Operations Director* l (GOD) yang ada juga di *file node-movement*. Objek "GOD" digunakan untuk menyimpan informasi global tentang keadaan dari lingkungan jaringan dan *node* di sekitarnya. Namun isi dari *file* "GOD" tidak boleh diketahui oleh setiap bagian dalam simulasi.

Dalam simulasi di sini objek "GOD" hanya digunakan untuk menyimpan sebuah *array* dari jumlah *hop* terpendek yang diperlukan untuk mencapai satu *node* ke *node* yang lain. Objek "GOD" tidak menghitung jumlah *hop* yang diperlukan selama simulasi berjalan, karena akan cukup memakan waktu. Namun "GOD" menghitung *hop* di akhir simulasi. Informasi yang dimuat ke dalam objek "GOD" dari pola pergerakan *file* terdapat pada baris perintah di Gambar 2.7.

```
$ns_ at 899.642 "$god_ set-dist 23 46 2"
```

Gambar 2.7 *Command Line* "GOD" pada 'scen-20-test'

Ini berarti bahwa jarak terpendek antara *node* 23 dan *node* 46 berubah menjadi 2 *hop* di waktu 899,642 detik. Program 'setdest' menghasilkan *file node-movement* menggunakan algoritma *random way point*. Perintah-perintah yang termasuk dalam program utama untuk memuat *file-file* ini dalam objek "GOD" [10].

2.6.2. *File Traffic-Connection Pattern*

Random traffic connection pada TCP dan CBR bisa dikonfigurasikan antara mobilitas *node* menggunakan skrip *traffic-scenario pattern generator*. Untuk menghasilkan alur *traffic* yang acak, dapat digunakan skrip Tcl yang disebut "cbrgen". Skrip ini membantu untuk menghasilkan *traffic load* atau beban trafik. Beban dapat berupa TCP atau CBR. Skrip ini

disimpan di dalam *file* 'CMU-scen-gen ' yang terletak di dalam direktori ~ns/indep-utils/cmu-scen-gen. Program "cbrgen.tcl" digunakan sesuai dengan *command line* pada Gambar 2.8 dan dengan keterangan yang ditunjukkan pada Tabel 2.2.

```
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed
seed] [-mc connections] [-rate rate] > traffic-
```

Gambar 2.8 Format *Command Line* cbrgen.tcl

Tabel 2.2 Keterangan *Command Line file* cbrgn.tcl

Parameter	Keterangan
-type cbr|tcp	Jenis *traffic* yang digunakan TCP atau CBR
-nn *node*s	Jumlah total *node*
-s seed	*Random seed*
-mc connections	Jumlah koneksi
-rate rate	Jumlah paket per detik. Pada CBR, panjang paket adalah tetap yaitu sebesar 512 bytes selama simulasi

Pada CBR, *data rate* dapat dihitung sebagai berikut:

Data Rate (bits/second) = 512 bytes*8 bits/bytes * rate (packets/second defined in "cbrgen")

Command line pada Gambar 2.9 membuat sebuah *file* koneksi CBR antara 50 *node*, yang memiliki maksimal 20 koneksi, dengan nilai *seed* 1.0 dan jumlah paket per detik sebanyak 4.0.

```
ns cbrgen.tcl -type cbr -nn 50 -seed 1.0 -mc 20 -
rate 4.0 > cbr-20-test
```

Gambar 2.9 Contoh *Command Line* cbrgen.tcl

Dari *file* cbr-20-test (kemana *output* dari generator akan diarahkan) sehingga menghasilkan salah satu koneksi CBR yang terlihat seperti pada Gambar 2.10.

```
#
# 2 connecting to 3 at time 82.557023746220864
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.25
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 82.557023746220864 "$cbr_(0) start"
```

Gambar 2.10 Koneksi CBR pada 'cbr-20-test'

Agent yang digunakan ialah UDP. Dengan demikian koneksi UDP adalah *setup* antara *node* 2 dan 3. Jumlah sumber UDP (dipilih antara *node* 0-50) dan jumlah *setup* koneksi diindikasikan sebagai 20 koneksi di akhir *file* cbr-20-test [10].

2.7. NS-2 *Trace File*

NS-2 *Trace File* merupakan *output* hasil *running* NS-2 yang berekstensi .tr. *Trace File* berisi *log* tentang semua jenis pengiriman dan penerimaan paket yang terjadi selama simulasi. Di dalam *Trace File* tercatat berbagai jenis paket sesuai jenis protokol *routing* yang digunakan. Tiap baris *log* ini dianalisis untuk mendapatkan performa protokol. Contoh pengiriman data paket pada NS-2 *Trace File* dapat dilihat pada Gambar 2.11.

```
s 26.000000000 _1_ AGT  --- 618 cbr 64 [0 0 0 0] -
------ [1:0 0:0 32 0] [26] 0 0
```

Gambar 2.11 *Trace* pengiriman paket data

Huruf "s" di kolom pertama menandakan pengiriman paket (*send*), kolom kedua berisi waktu pengiriman paket pada detik ke 26, kolom ketiga merupakan *node* tempat *event* terjadi yaitu pada *node* 1, kolom ke empat berisi AGT yang menandakan pengiriman paket data, kolom ke lima merupakan tempat terjadinya *event* spesial semisal *collision*, kolom ke 6 merupakan id unik paket, kolom ke tujuh berisi tipe paket yang dikirimkan yaitu cbr, kolom ke delapan merupakan ukuran paket dalam *byte* yaitu 64.

Untuk penerimaan data paket seperti berikut tidak jauh beda dengan pengiriman data paket, pembedanya yaitu kolom pertama dengan huruf inisial "r" yang menandakan paket diterima dan format selanjutnya sama persis dengan paket pengiriman. Contoh penerimaan data paket pada NS-2 *Trace File* dapat dilihat pada Gambar 2.12.

```
r 26.011627928 _0_ AGT  --- 618 cbr 84 [13a 0 16
800] ------- [1:0 0:0 27 0] [26] 5 0
```

Gambar 2.12 *Trace* Penerimaan Paket Data

Contoh format untuk paket *routing* OLSR pada *trace file* seperti pada Gambar 2.13.

```
r 30.334151244 _12_ RTR  --- 741 OLSR 64 [0
ffffffff 18 800] ------- [24:255 -1:255 32 0] [1
17 [HELLO 24 0 17]]
```

Gambar 2.13 *Trace* Pengiriman Paket *Routing* OLSR

2.8. Awk

Awk adalah sebuah pemrograman seperti pada *shell* atau C yang memiliki karakteristik yaitu sebagai *tools* yang cocok filter / manipulasi. Awk adalah penggabungan dari nama lengkap sang author, yaitu : Alfred V. Aho, Peter J. Weinberger dan Brian W. Kernighan. Awk atau juga disebut Gawk (GNU awk), yaitu bahasa pemrograman umum dan *utility* standard POSIX 1003.2. Jika kecepatan merupakan hal yang penting, awk adalah bahasa yang sangat sesuai. Awk sangat baik untuk manipulasi *file* teks.

Secara umum bahasa pemrograman awk dapat digunakan untuk mengelola database sederhana, membuat laporan, memvalidasi data, menghasilkan indeks dan menampilkan dokumen, membuat algoritma yang digunakan untuk mengubah bahasa komputer ke bahasa lainnya. Dengan kata lain awk menyediakan fasilitas yang dapat memudahkan untuk memecah bagian data untuk proses selanjutnya, mengurutkan data dan menampilkan komunikasi jaringan yang sederhana.

Fungsi dasar awk adalah untuk mencari *file* per baris (atau unit teks lain) yang berisi pola tertentu. Ketika suatu baris sesuai dengan pola, awk melakukan aksi yang khusus pada baris tersebut. awk tetap memproses baris *input* sedemikian hingga mencapai akhir baris *input*. Program pada awk berbeda dari program di kebanyakan bahasa lain, karena program awk bersifat "data-driven" yang mana diperlukan pendeskripsian data yang dikehendaki untuk bekerja dan kemudian apa yang akan dilakukan saat data tersebut ditemukan. Kebanyakan bahasa lainnya bersifat "procedural" maka dari itu diharuskan mendeskripsikannya secara detail setiap langkah program yang harus dijalankan. Ketika bekerja dengan bahasa prosedural, biasanya sangat sulit untuk mendeskripsikan data yang hendak diproses oleh program. Oleh karena itu, program awk sering kali terasa lebih mudah untuk ditulis dan dibaca [11].

Pada tugas akhir ini AWK digunakan untuk membuat *script* dalam penghitungan *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2D), dan *Routing Overhead* (RO) dari hasil *trace* NS-2.

[Halaman ini sengaja dikosongkan]

BAB III
PERANCANGAN SISTEM

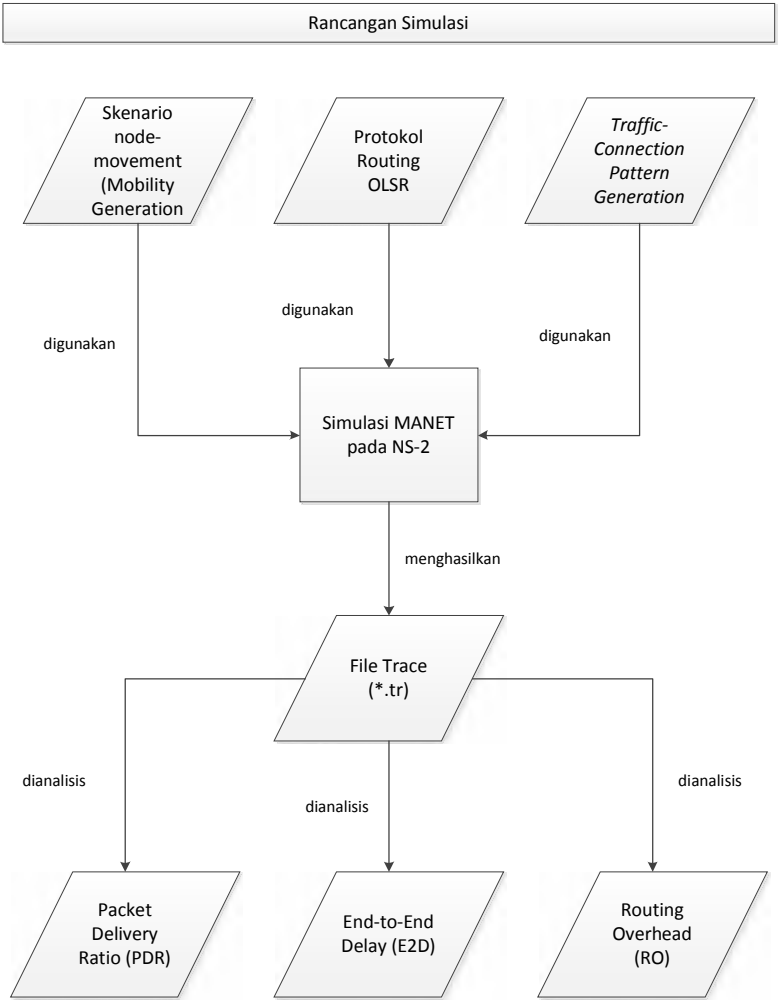
Pada bab ini dijelaskan mengenai dasar perancangan dari perangkat lunak yang akan dibangun dalam Tugas Akhir ini. Secara khusus akan dibahas mengenai deskripsi umum sistem, perancangan skenario, alur, serta gambaran implementasi sistem yang diterapkan pada *Network Simulator* 2 (NS-2).

3.1. Deskripsi Umum

Pada Tugas Akhir ini akan dilakukan analisis tentang performa model tranmsisi *TwoRayGround* dan Nakagami pada MANET. Dalam pembuatan skenario MANET menggunakan *Mobility Generator* yang bersifat *Random Way Point* dan telah ada pada *Network Simulator*-2 (NS-2) yaitu dengan cara men-*generate file node-movement (mobility generation)* dan membuat koneksi antar *node* menggunakan *file traffic-connection pattern*. Rancangan Simulasi yang dibuat dapat dilihat pada Gambar 3.1.

Dalam penelitian ini, terdapat 2 jenis model transmisi yang digunakan sebagai perbandingan pengukuran performa dari *mobility generator* yaitu model transmisi *TwoRayGround* dan Nakagami. Kemudian untuk simulasi skenario yang dihasilkan oleh *mobility generator* akan dijalankan pada NS-2 menggunakan protokol *routing* OLSR dengan model transmisi yang berbeda pada sistem operasi Linux.

Pada tiap skenario, kecepatan maksimum pergerakan dari satu *node* ke *node* lainnya dibuat bervariasi yaitu 5, 10 dan 15 m/s. Hasil proses uji coba dari tiap skenario akan menghasilkan sebuah *trace file* yang nantinya akan dilakukan analisis perhitungan metrik *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2D), dan *Routing Overhead* (RO). Dari hasil metrik tersebut dianalisis performa kedua model transmisi yang dibandingkan.



Gambar 3.1 Tahapan Rancangan Simulasi

3.2. Perancangan Skenario

Perancangan skenario uji coba mobilitas MANET diawali dengan melakukan pembuatan skenario pada *mobility generation* yang bersifat *Random Way Point* kemudian membuat koneksi

dengan menggunakan *file traffic-connection* yang sudah ada pada NS-2. Pada Tugas Akhir ini pembuatan skenario untuk melihat pergerakan *node* dibedakan berdasarkan 3 kecepatan maksimum yaitu 5, 10 dan 15 m/s. Sedangkan untuk koneksinya hanya digunakan 2 *node* untuk menentukan *node* pengirim dan *node* penerima paket. Penjelasan untuk perancangan skenario pada *mobility generator* dan pembuatan koneksi antar *node*-nya adalah sebagai berikut:

3.2.1. Skenario *Node-Movement (Mobility Generation)*

Skenario *mobility generation* dibuat dengan men-*generate file node-movement* yang telah ada pada NS-2 atau *tools*-nya biasa disebut 'setdest' yang nantinya akan menghasilkan *output* dalam bentuk .txt dan digunakan dalam *file* Tcl selama simulasi pada NS-2 sebagai bentuk pergerakan *node* yang berpindah-pindah.

Tabel 3.1 Parameter Skenario *Node-Movement*

No.	Parameter	Spesifikasi
1	Jumlah *Node*	50
2	Waktu Simulasi	100 detik
3	Area	510 m x 510 m
3	Kecepatan Maksimal	- 5 m/s - 10 m/s - 15 m/s
5	Sumber *Traffic*	CBR
6	Waktu Jeda (dalam detik)	10
7	Ukuran Paket	512 bytes
8	*Rate* Paket	0.25 paket per detik
9	Jumlah maksimal koneksi	1
10	Model mobilitas yang digunakan	*Random Way Point*

3.2.2. *Traffic-Connection Pattern Generation*

Traffic-Connection dibuat dengan menjalankan program cbgren.tcl yang telah ada pada NS-2 yang nantinya akan menghasilkan *output* dalam bentuk .txt dan digunakan sebagai koneksi untuk menghubungkan antar *node* yang ada pada skenario selama simulasi pada NS-2.

Tabel 3.2 Parameter *Traffic-Connection Pattern*

No.	Parameter	Spesifikasi
1	-type cbr|tcp	CBR
2	-nn *node*s	2
3	-s seed	1.0
4	-mc connections	1
5	-rate rate	0.25
6	*Agent*	UDP

3.3. Perancangan Simulasi pada NS-2

Pada perancangan kode NS-2 dengan konfigurasi MANET, dilakukan penggabungan skenario mobilitas dan *traffic-connection* dengan skrip TCL yang diberikan parameter-parameter untuk membangun percobaan simulasi MANET pada NS-2. Berikut parameter simulasi perancangan sistem MANET yang dapat digunakan dapat dilihat pada Tabel 3.3.

Tabel 3.3 Parameter Simulasi pada NS-2

No.	Parameter	Spesifikasi
1	Network simulator	NS- 2.35
2	*Routing* Protocol	OLSR
3	Waktu simulasi	100 detik
4	Waktu Pengiriman Paket Data	- *TwoRayGround* = 0 – 100 detik - Nakagami = 0 – 100 detik
5	Area simulasi	510 m x 510 m

No.	Parameter	Spesifikasi
6	Banyak *node*	50
7	Radius transmisi	100 m
8	Tipe koneksi	UDP
9	Tipe data	Constant Bit Rate (CBR)
10	Source / Destination	Statik (*Node* 1 / *Node* 2)
11	Kecepatan generasi paket	1 paket per detik
12	Ukuran paket data	512 bytes
13	Protokol MAC	IEEE 802.11
14	Mode Transmisi	- *TwoRayGround* - Nakagami
15	Tipe Antena	OmniAntenna
16	Tipe Interface Queue	Droptail/PriQueue
17	Tipe Peta	MANET (*random way point*)
18	Tipe kanal	Wireless channel
19	Tipe *trace*	Old Wireless Format *Trace*

3.4. Perancangan Metrik Analisis

Metrik yang akan dianalisis pada Tugas Akhir ini adalah *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2D), dan *Routing Overhead* (RO). Penjelasannya sebagai berikut:

3.4.1. *Packet Delivery Ratio* (PDR)

PDR dihitung dari perbandingan antara paket yang dikirim dengan paket yang diterima. PDR dihitung dengan menggunakan persamaan (4), dimana *received* adalah banyaknya paket data yang diterima dan *sent* adalah banyaknya paket data yang dikirimkan.

$$P = \frac{r}{s} \times 100$$

(4)

3.4.2. *End-to-End Delay* (E2D)

E2D dihitung dari rata-rata *delay* antara waktu paket diterima dan waktu paket dikirim. E2D dihitung dengan menggunakan persamaan (5), dimana $t_{received[i]}$ adalah waktu penerimaan paket dengan urutan / id ke-i, $t_{sent[i]}$ adalah waktu pengiriman paket dengan urutan / id ke-i, dan *sent* adalah banyaknya paket data yang dikirimkan.

$$E2D = \frac{\sum_{i=0}^{i \le s} t_{r\ [i]} - t_{s\ [i]}}{s}$$

(5)

3.4.3. *Routing Overhead* (RO)

Routing Overhead adalah jumlah paket kontrol *routing* yang ditransimisikan per data paket yang terkirim ke tujuan selama simulasi terjadi. RO dihitung berdasarkan jumlah paket *routing* yang ditransmisikan. Baris yang mengandung *routing overhead* pada *trace file* ditandai dengan paket yang bertipe *send* (s) / *forward* (f) dan terdapat *header* paket dari protokol OLSR.

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan perangkat lunak. Implementasi yang dijelaskan meliputi lingkungan pembangunan perangkat lunak, implementasi skenario, implementasi simulasi pada NS-2, dan implemantasi matrik analisis.

4.1. Lingkungan Pembangunan Perangkat Lunak

Pembangunan perangkat lunak dilakukan pada lingkungan pengembangan sebagai berikut:

4.1.1. Lingkungan Perangkat Lunak

Adapun perangkat lunak yang digunakan untuk pengembangan sistem adalah sebagai berikut:

- Sistem Operasi Ubuntu 10.04 LTS 64 bit untuk lingkungan NS-2;
- *Network Simulator* 2 versi 2.35.
- *Patch* OLSR versi 1.0

4.1.2. Lingkungan Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk implementasi perangkat lunak Tugas Akhir adalah sebagai berikut:

- *Processor* Intel(R) Core(TM) i5-2450M CPU @2.50GHz;
- Media penyimpanan sebesar 500GB;
- RAM sebesar 4 GB DDR3.

4.2. Implementasi Skenario

Implementasi skenario mobilitas MANET dipelajari dalam kondisi yang berbeda pada beban *traffic*-nya dan mobilitas/pergerakan *node*-nya. Dua model yang digunakan untuk studi simulasi pada jaringan MANET adalah *mobility* generation, yang digunakan untuk mempelajari mempelajari pengaruh mobilitas dari *node* pada kinerja keseluruhan jaringan dan *traffic*-

connection generation, yang digunakan untuk mempelajari pengaruh beban *traffic* pada jaringan. Implementasi skenarionya adalah sebagai berikut:

4.2.1. Skenario *File Node-Movement (Mobility Generation)*

Dalam implementasi skenario pada *mobility generation* menggunakan *tools generate default* yang dimiliki oleh NS-2 yaitu 'setdest'. *File* skenario *node-movement (mobility generation)* digunakan untuk setiap simulasi yang ditandai dengan jeda waktu. Untuk mempelajari efek mobilitas, simulasi dilakukan dengan pola gerakan yang dihasilkan dari kecepatan maksimal yang berbeda. *Setting* pada *file* skenario pergerakan *node* sesuai dengan mobilitas yang berbeda, dibuat dengan memvariasikan kecepatan maksimal. Program 'setdest' pada NS-2 digunakan untuk menghasilkan *file node-movement* dengan menggunakan algoritma *Random Way Point*. Format *command line* pada Gambar 4.1 yang digunakan untuk menghasilkan gerakan acak pada *node* adalah sebagai berikut:

```
./setdest [-v version ] [-n num_of_nodes] [-p
pausetime] [-M maxspeed] [-t simtime] [-x maxx] [-
y maxy] > [outdir/movement-file]
```

Gambar 4.1 Format *Command Line* 'setdest'

Ketentuan-ketentuan yang diujicobakan pada skenarionya berturut-turut adalah versi 'setdest' simulator yaitu 1, jumlah *node* dalam skenario yaitu 50, waktu jeda (*pause time*) yaitu 10 detik, kecepatan maksimalnya yaitu skenario A sebesar 5 m/s, skenario B sebesar 10 m/s dan skenario C sebesar 15 m/s, waktu simulasi yaitu 100 detik, panjang dan lebar maksimal area simulasi yaitu 510 meter. Kemudian *file* mobilitas yang dihasilkan disimpan dalam direktori " ~ ns /indep-utils/CMU-scen-gen/setdest / ". Pada Gambar 4.2 dapat dilihat implementasi *command line* pada 'setdest' dengan berbagai kecepatan maksimal yang berbeda sesuai dan *node* sebanyak 50. Dan untuk

setiap kecepatan maksimal tersebut dibuat 10 buah *file* untuk satu protokol *routing* dan satu model transmisi.

```
./setdest –v 1 –n 50 –p 10 –M 5 –t 100 –x 510 –y
510 > scena1.txt
./setdest –v 1 –n 50 –p 10 –M 10 –t 100 –x 510 –y
510 > scenb1.txt
./setdest –v 1 –n 50 –p 10 –M 15 –t 100 –x 510 –y
510 > scenc1.txt
```

Gambar 4.2 Implementasi pada 'setdest'

Pada Gambar 4.3 terdapat potongan dari *file* mobilitas scena1.txt hasil *generate* 'setdest' yang menunjukkan penempatan awal setiap *node* dalam sebuah area dan dinyatakan dalam sumbu X, Y dan Z adalah sebagai berikut:

```
#
# nodes: 50, pause: 10.00, max speed: 5.00, max x:
510.00, max y: 510.00
#
$node_(0) set X_ 448.902173976333
$node_(0) set Y_ 416.404343429511
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 473.407075320993
$node_(1) set Y_ 226.655608269937
```

Gambar 4.3 Posisi *Node* dalam X, Y dan Z

Sedangkan untuk Gambar 4.4 menunjukkan pergerakan *node* tersebut selama waktu simulasi dijalankan untuk setiap *node* diberikan posisi awal dan berkelanjutan untuk menentukan pergerakan *node* berikutnya. Dari potongan dari scena1.txt pada Gambar 4.4, baris pertama mendefinisikan untuk *node* (0) pada detik ke 10 mulai bergerak ke arah tujuan (191.73, 125. 21) dengan kecepatan 0.34 m/s. Baris perintah ini dapat digunakan untuk mengubah arah dan kecepatan pada pergerakan mobilitas *node*.

```
$ns_ at 10.000000000000 "$node_(0) setdest
191.734433653942 125.212223281651 0.345476796352"
$ns_ at 10.000000000000 "$node_(1) setdest
125.972955560399 280.372260063288 3.433236813665"
$ns_ at 10.000000000000 "$node_(2) setdest
304.565395484586 296.333151906099 2.639912491222"
```

Gambar 4.4 Perpindahan/Pergerakan *Node*

Kemudian pada Gambar 4.5 terdapat potongan dari scena1.txt yang menunjukkan penentuan GOD untuk setiap *node*. GOD sendiri merupakan kepanjangan dari *General Operations Director*, yang berguna untuk menyimpan informasi global tentang jumlah dan pergerakan *node*. Saat ini, objek GOD hanya digunakan untuk menyimpan *array* terpendek dari *hop* yang diperlukan untuk mencapai dari satu *node* ke *node* yang lain. Objek GOD tidak menghitungan *array* ini selama simulasi berjalan karena bisa memakan waktu.

```
$god_ set-dist 0 1 1
$god_ set-dist 0 2 1
$god_ set-dist 0 3 2
$god_ set-dist 0 4 1
$god_ set-dist 0 5 3
$god_ set-dist 0 6 1
```

Gambar 4.5 Pembuatan GOD untuk Setiap *Node*

Lalu informasi yang dimuat ke dalam objek GOD dari *file node-movement* ditunjukkan pada Gambar 4.6. Pada baris pertama dari potongan *file* scena1.txt tersebut, digunakan untuk memuat objek GOD dengan informasi yaitu jalan terpendek antara *node* 6 dan *node* 20 berubah menjadi 2 *hop* pada detik ke-10.180742080271.

```
$ns_ at 10.180742080271 "$god_ set-dist 6 20 2"
$ns_ at 10.180742080271 "$god_ set-dist 20 34 1"
$ns_ at 10.302439202349 "$god_ set-dist 2 14 1"
```

Gambar 4.6 *Access Point*

4.2.2. *File Traffic-Connection Pattern Generation*

Dalam implementasi *random traffic-connection generation* untuk TCP dan CBR dapat di setting dengan pergerakan antar *node* menggunakan skrip *traffic-scenario generator*. Skrip *traffic generator* ini terdapat pada direktori ~ns/indep-utils/cmu-scen-gen dan disimpan dalam bentuk *file* cbrgen.tcl. *File* ini dapat digunakan untuk membuat *traffic-conneciton* CBR ataupun TCP pada jaringan pergerakan antar *node*. Pada saat menjalankan perintah pada *file traffic-connection* cbrgen.tcl ini, kita harus mendefinisikan kan tipe *traffic-connection*-nya (CBR atau TCP), banyaknya *node* dan koneksi maksimal yang ada pada jaringan tersebut, *random seed* dan misalkan pada koneksi CBR, *rate* yang nilai kebalikannya digunakan untuk menghitung waktu interval antar paket CBR yang kemudian disimpan dalam sebuah *file traffic*. Format *command line* pada Gambar 4.7 yang digunakan untuk menghasilkan gerakan acak pada *node* adalah sebagai berikut:

```
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed
seed] [-mc connections] [-rate rate] > traffic-
```

Gambar 4.7 Format *Command Line* cbrgen.tcl

Waktu awal untuk koneksi TCP/CBR secara acak yang dihasilkan dengan nilai maksimal ditetapkan pada 180.0 detik. Pada Gambar 4.8 merupakan bentuk implementasi untuk menjalankan cbrgent.tcl untuk membuat *file* koneksi CBR diantara 2 *node*, memiliki maksimal 1 koneksi dengan nilai *seed* 1.0 dan jumlah paket per detik sebanyak 0.25 yang disimpan dalam cbrtest.txt yang nantinya akan digunakan pada saat simulasi NS-2.

```
ns cbrgen.tcl CBR –nn 2 –seed 1.0 –mc 1 –rate 0.25
> cbrtest.txt
```

Gambar 4.8 Implementasi Koneksi cbrgen.tcl

Kemudian Gambar 4.9 menunjukkan *output*-nya yang disimpan dalam cbrtest.txt sehingga menghasilkan koneksi CBR

dan menggunakan *Agent* UDP. Koneksi UDP di sini merupakan konfigurasi antara *node* ke-1 dan 2. Interval pengiriman paket data dilatkukan setiap satu detik dengan besar paket data 512 byte dan maksimal pengiriman paket 10.000.

```
#
# nodes: 2, max conn: 1, send rate: 0.25, seed:
1.0
#
# 1 connecting to 2 at time 2.5568388786897245
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 1
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
```

Gambar 4.9 *Output* cbrtest.txt

4.3. Implementasi Simulasi pada NS-2

Untuk mensimulasikan MANET dalam lingkungan NS-2, skrip tcl yang telah ada pada *pacth* protokol *routing* OLSR. Untuk Skenario *node-movement (mobility generation)* dan *traffic-connection generation* dan yang disimpan dalam bentuk .txt diberikan parameter-parameter yang sesuai dengan perancangan agar dapat dijalankan pada NS-2 . Parameter-parameter tersebut dibuat menggunakan bahasa Tcl/Otcl.

Pada Gambar 4.10 menunjukkan skrip konfigurasi awal parameter-parameter yang diberikan untuk menjalankan MANET pada NS-2. Baris pertama merupakan konfigurasi tipe *channel* yang digunakan yaitu *Wireless Channel*. Baris kedua merupakan tipe transmisi yang digunakan karena pada Tugas Akhir ini membanding dua model transmisi maka pada baris kedua ini bisa diganti dengan model transmisi *TwoRayGround* atau Nakagami

untuk pengujian simulasi. Tipe Mac yang digunakan adalah Mac 802.11. Pada baris diatas juga dilakukan konfigurasi tipe *queue* dari *interface*, tipe *link layer*, tipe *antenna* dan jumlah maksimal *packet* pada *interface queue*. Baris ke-9 sampai baris ke-17 berturut-turut merupakan koordinat x serta koordinat y sebesar 510 meter, jumlah paket dan *node* yaitu 50 *node*, protokol *routing* yang digunakan yaitu OLSR, besarnya *seed* yaitu 0.0, waktu simulasi diakhiri pada detik ke-100, *file traffic-connection* yang digunakan yaitu cbrtest.txt dan terakhir ialah *file* skenario *node-movement (mobility generation)* yang digunakan adalah scena1.txt.

```
set val(chan)   Channel/WirelessChannel;
set val(prop)   Propagation/TwoRayGround;
set val(netif)  Phy/WirelessPhy;
set val(mac)    Mac/802_11;
set val(ifq)    Queue/DropTail/PriQueue;
set val(ll)     LL;
set val(ant)    Antenna/OmniAntenna;
set opt(x)      510;
set opt(y)      510;
set val(ifqlen) 50;
set val(nn)     50;
set val(seed)   0.0;
set val(rp)     OLSR;
set val(stop)   100
set val(cp)     “cbrtest.txt”;
set val(sc)     “scena1.txt”;
```

Gambar 4.10 Konfigurasi awal parameter NS-2

Pada Gambar 4.11 merupakan pengaturan dari *transmission range* yang digunakan pada simulasi. Nilai yang diubah ialah *RXThresh_* (*Receiver Sensitivity Threshold*). Nilai `1.42681e-08` pada variabel tersebut memiliki artian bahwa *range* atau jangkauan yang dapat dicapai ialah sejauh 100 meter.

```
Phy/WirelessPhy set RXThresh_ 1.42681e-08
```

Gambar 4.11 Konfigurasi *Transmission Range* pada NS-2

```
# Initialize Global Variables
# create simulator instance
set ns_      [new Simulator]

# create trace object for ns and nam
set tracefd  [open olsr_outtrga10.tr w]
set namtrace  [open olsr_outtrga10.nam w]

$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x)
$opt(y)
# set up topology object
set topo    [new Topography]
$topo load_flatgrid $opt(x) $opt(y)

# Create God
set god_ [create-god $val(nn)]

# Configure node
$ns_ node-config  -adhocRouting $val(adhocRouting)
\
               -llType $val(ll) \
               -macType $val(mac) \
               -ifqType $val(ifq) \
               -ifqLen $val(ifqlen) \
               -antType $val(ant) \
               -propType $val(prop) \
               -phyType $val(netif) \
               -channelType $val(chan) \
               -topoInstance $topo \
               -agentTrace ON \
               -routerTrace ON \
               -macTrace OFF \
               -movementTrace ON \

for {set i 0} {$i < $val(nn)} {incr i} {
  set node_($i) [$ns_ node]
  $node_($i) random-motion 0 ;# disable random
motion
}
```

Gambar 4.12 Konfigurasi *Trace File*, NAM dan Pergerakan *Node* pada NS-2

Skrip yang ditunjukkan pada Gambar 4.12 merupakan skrip untuk pengaturan variabel global yang diawali dengan *set ns* untuk pembuatan simulator baru. *Set tracefd* dan *set namtrace* merupakan pengaturan untuk nama *trace file* berekstensi .tr dan *file network* animator .nam akan dihasilkan dan disimpan.

Pada *set tracefd* dapat dilakukan pengaturan untuk menghasilkan *trace file* sesuai dengan keinginan pengguna. Terdapat dua jenis format *trace file* yang disediakan yaitu *old trace format* dan *new trace format* namun pada Tugas Akhir ini digunakan *old trace format*. *Set topo* merupakan pengaturan untuk objek topografi berdasarkan pada luas koordinat yang telah dikonfigurasi sebelumnya. *Create-god* dan *node-config -channelType* merupakan konfigurasi yang dilakukan pada *node-node* yang akan dibuat. Pada *create-god* dilakukan implementasi *node-node* yang akan dibuat sesuai dengan parameter pada *set-val(nn)* sedangkan pada *node-config -channelType* merupakan konfigurasi *node* sesuai dengan parameter-parameter yang telah ditambahkan sebelumnya pada Gambar 4.10 seperti tipe *link layer*, tipe mac dan tipe transmisi. Terakhir dilakukan perulangan untuk membuat pergerakan dari *node-node*. *Node-node* yang dibuat tidak dapat melakukan pergerakan secara acak karena pergerakan *node* merupakan *trace file* yang dihasilkan oleh *mobility generator*.

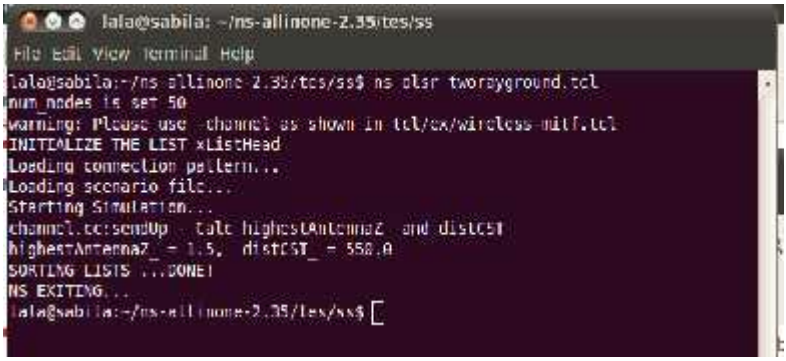
Skrip pada Gambar 4.13 merupakan bagian akhir dari keseluruhan skrip yang digunakan untuk menginisialisasi penempatan awal *node-node* yang dibut pada skenario *node-movement (mobility generation)*, pergerakan *node* tersebut selama waktu simulasi dilakukan dan melakukan konfigurasi pengiriman paket data yang dilakukan yang nantinya dihasilkan pada *file output* .tr. Pada potongan skrip tersebut, akan dipanggil *file* skenario *node-movement (mobility generation)* dan *traffic-connection pattern* kemudian pengiriman paket data dimulai pada detik ke-0 dan diberhentikan pada detik ke-100 seperti yang telah dikonfigurasi sebelumnya pada Gambar 4.10.

```
# Define node movement model
puts "Loading connection pattern..."
source $val(cp)

# Define traffic model
puts "Loading scenario file..."
source $val(sc)

# Define node initial position in nam

for {set i 0} {$i < $val(nn)} {incr i} {

    # 20 defines the node size in nam, must adjust
it according to your scenario
    # The function must be called after mobility
model is defined

    $ns_ initial_node_pos $node_($i) 20
}

# Tell nodes when the simulation ends
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop).0 "$node_($i) reset";
}

$ns_ at  $val(stop).0002 "puts \"NS EXITING...\" ;
$ns_ halt"

puts $tracefd "M 0.0 nn $val(nn) x $opt(x) y
$opt(y) rp $val(adhocRouting)"
puts $tracefd "M 0.0 sc $val(sc) cp $val(cp) seed
$val(seed)"
puts $tracefd "M 0.0 prop $val(prop) ant
$val(ant)"

puts "Starting Simulation..."
$ns_ run
```

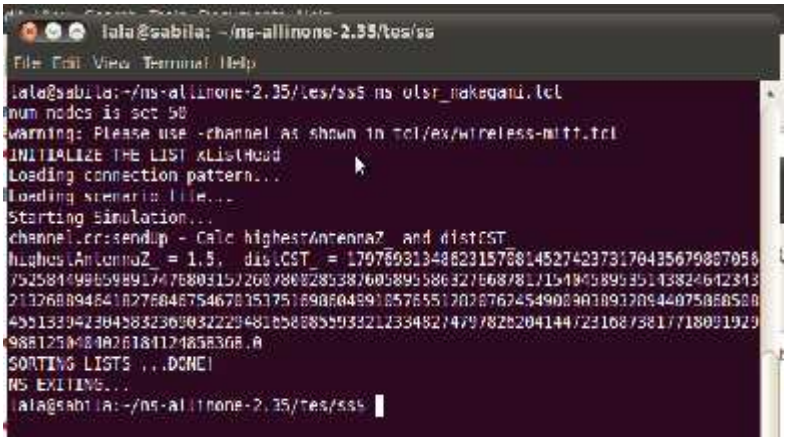
Gambar 4.13 Konfigurasi pengiriman paket data NS-2

Pada Gambar 4.14 merupakan contoh *running*/eksekusi *file* .tcl menggunakan model transmisi *TwoRayGround* dan Gambar 4.15 untuk model transmisi Nakagami dengan jumlah

node 50 yang bersifat acak atau *Random Way Point* sesuati dengan skenario *node-movement* yang telah di-*generate* sebelumnya.

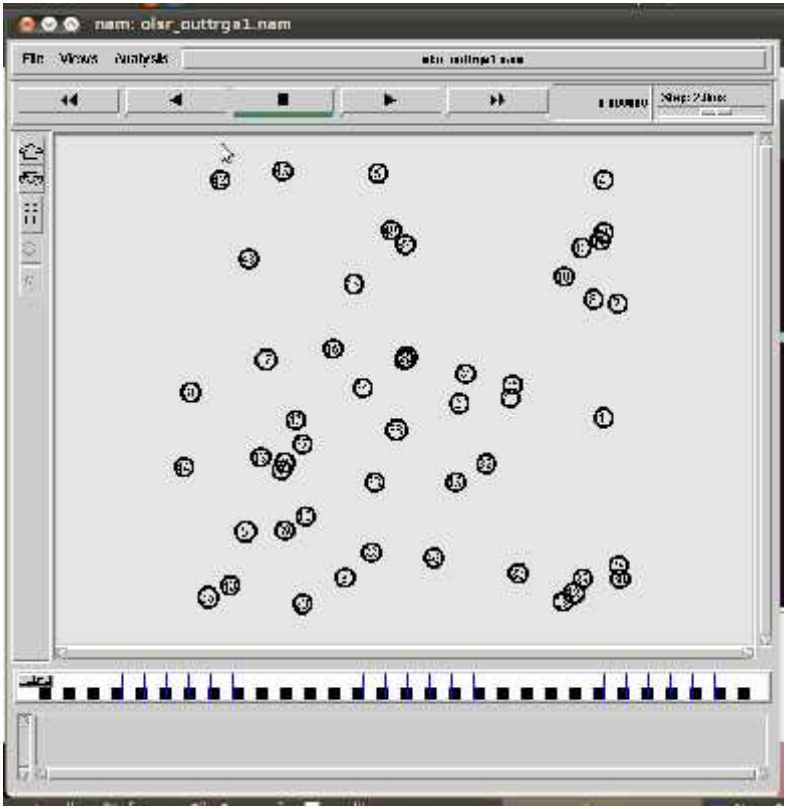


Gambar 4.14 Perintah Eksekusi Model Transmisi *TwoRayGround*



Gambar 4.15 Perintah Eksekusi Model Transmisi Nakagami

Hasil yang diperoleh dari *running*/eksekusi berkas .tcl berupa *trace file* berbentuk .tr dan *file* animasi pengiriman paket data berbentuk .nam. *File* .tr inilah yang akan dianalisis parameter-parameternya berupa PDR, *delay*, dan RO. Pada Gambar 4.16 menunjukkan tampilan hasil dari *file* .nam yang dihasilkan.



Gambar 4.16 Visualisasi hasil simulasi pada NAM

4.4. Implementasi Metrik Analisis

Hasil menjalankan skenario VANET dalam NS-2 dalam bentuk *Trace File* berekstensi .tr dianalisis dengan 3 (tiga) metrik yaitu *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2D), dan *Routing Overhead* (RO). Implementasi dari tiap metrik menggunakan bahasa pemrograman AWK dan dijelaskan seperti berikut.

4.4.1. *Packet Delivery Ratio* (PDR)

Proses perhitungan PDR dilakukan dengan menghitung jumlah paket data terkirim yang dilakukan oleh *node* 1 dan jumlah paket data yang diterima oleh *node* 2 pada satu *trace file*. Penambahan jumlah paket terkirim dilakukan apabila pada baris *trace* yang bersangkutan mengandung semua kondisi dimana kolom pertama mengandung huruf "s" yang menandakan *send packet*, kolom ke-3 menunjukkan bahwa *node* yang melakukan pengiriman adalah *node* 1, kolom ke-4 dan kolom ke-7 mengandung huruf "AGT" dan "cbr" yang menandakan pengiriman paket yang dilakukan adalah pengiriman paket data. Pencatatan jumlah paket yang diterima dilakukan apabila pada baris *trace* yang bersangkutan mengandung semua kondisi dimana kolom pertama mengandung huruf "r" yang menandakan *received packet*, kolom ke-3 menunjukkan bahwa *node* yang menerima packet data adalah *node* 2, kolom ke-4 dan kolom ke-7 mengandung huruf "AGT" dan "cbr" yang menandakan penerimaan paket yang diterima adalah paket data. Perhitungan dilakukan sampai baris terakhir *trace file*, dan hasilnya adalah hasil hitung nilai PDR simulasi skenario.

Psedeucode PDR ditunjukkan pada Gambar 4.17 dan implementasinya dapat dilihat pada Lampiran.

```
ALGORITMA PDR(trace file)
//Input: trace file simulasi skenario
//Ouput: jumlah packet sent, packet received, dan
//       PDR
BEGIN (
sent ← 0
recv ← 0
recv_id ← 0
pdr ← 0)

(if ($1 == "s" and $3 == "_1_" and $4 == "AGT" and
      $7 == "cbr" )
      sent + 1;
```

```
if ($1 == "r" and $3 == "_2_" and $4 == "AGT" and
       $7 == "cbr")
       recv +1;
)
END (
pdr ← ( recv / sent ) * 100
print sent
print recv
print pdr)
```

Gambar 4.17 *Pseudeucode* PDR

Contoh perintah untuk analisis PDR dari *trace file* model transmisi *TwoRayGround* dengan protokol OLSR dan kecepatan maksimal perpindahan *node* sebesar 5 m/s seperti pada Gambar 4.18.

```
awk -f pdr.awk olsr_outtrga1.tr
```

Gambar 4.18 Perintah Menjalankan Skrip pdr.awk

Contoh *output* dari menjalankan skrip tersebut ditunjukkan pada Gambar 4.19.

```
Transmitted packet(s) =  99
Received packet(s) =  51
Packet Delivery Ratio =  51.5152 %
```

Gambar 4.19 Hasil *Running* skrip pdr.awk

4.4.2. *End-to-End Delay* (E2D)

Perhitungan E2D dilakukan dengan menghitung selisih waktu paket data terkirim yang dilakukan oleh *node* 1 dan waktu paket data diterima oleh *node* 2 di dalam satu *trace file*. Pencatatan waktu paket terkirim pada kolom ke-2 dilakukan apabila pada baris *trace* yang bersangkutan mengandung semua kondisi yaitu kolom pertama mengandung huruf “s” yang menandakan *send packet*, kolom ke-3 menunjukkan *node* yang melakukan pengiriman adalah *node* 1, kolom ke-4 dan kolom ke-7 mengandung huruf “AGT” dan “cbr” yang menunjukkan pengiriman paket yang dilakukan adalah pengiriman paket data.

Perhitungan waktu dan pencatatan ID serta jumlah paket diterima dilakukan apabila baris *trace* yang bersangkutan mengandung kondisi dimana yaitu kolom pertama mengandung huruf "r" yang menandakan *received packet*, kolom ke-3 menunjukkan *node* yang menerima paket adalah *node* 2, kolom ke-4 dan kolom ke-7 mengandung huruf "AGT" dan "cbr" yang menunjukkan penerimaan paket yang adalah penerimaan paket data. Perhitungan dilakukan sampai baris terakhir *trace file*, dan dilakukan perhitungan nilai E2D dengan menghitung selisih *delay* paket mulai dari pengiriman sampai paket diterima pada simulasi skenario. *Psedeucode* E2D ditunjukkan pada Gambar 4.20 dan implementasinya dapat dilihat pada Lampiran.

```
ALGORITMA E2D(trace file)
//Input: trace file simulasi skenario
//Ouput: jumlah packet receieved, total delay, dan
//        E2D
BEGIN (
for i in pkt_id
pkt_id[i] ← 0
for i in pkt_sent
       pkt_sent[i] ← 0
for i in pkt_recv
       pkt_recv[i] ← 0
delay = avg_delay ← 0
recv ← 0
recv_id ← 0)

(if ($1 == "s" and $3 == "_1_" and $4 == "AGT" and
     $7 == "cbr)
      pkt_sent[$6] ← $2

if ($1 == "r" and $3 == "_2_" and $4 == "AGT" and
    $7 == "cbr" and  recv_id != $6 )
              recv + 1
recv_id ← $6
              pkt_recv[$6] ← $2;
)
```

```
END (
for i in pkt_recv
       delay += pkt_recv[i] - pkt_sent[i]
avg_delay ← delay / recv;
print  recv
print delay
print avg_delay
)
```

Gambar 4.20 *Pseudeucode* E2D

Contoh perintah untuk analisis E2D dari *trace file* model transmisi *TwoRayGround* dengan protokol OLSR dan kecepatan maksimal perpindahan *node* sebesar 5 m/s seperti Gambar 4.21 dan hasilnya dapat dilihat pada Gambar 4.22.

```
awk -f endtoend.awk olsr_outtrga1.tr
```

Gambar 4.21 Perintah Menjalankan skrip endtoend.awk

```
Total Packet(s) Receive = 51
Total Delay = 0.605506 second
Average Packet Delivery Delay =  0.0118727 second
```

Gambar 4.22 Hasil *Running* skrip endtoend.awk

4.4.3. *Routing Overhead* (RO)

Implementasi perhitungan metrik *Routing overhead* OLSR dihitung apabila kondisi-kondisi yang ada terpenuhi yatu pada kolom pertama diawali dengan huruf "s" yang berarti *send packet* atau huruf "f" yang berarti *forward packet*, kolom ke-4 mengandung huruf "RTR" yang berarti paket *routing* dan kolom ke-7 mengandung "OLSR" yang berarti paket *routing* OLSR. seperti pada Gambar 4.23. Implementasinya dapat dilihat pada Lampiran.

```
ALGORITMA RO-OLSR(trace file)
//Input: trace file simulasi skenario
//Ouput: jumlah routing overhead protokol OLSR
BEGIN (
rt_pkts ← 0)
(if (($1=="s" || $1=="f") && $4 == "RTR" &&
       $7 == "OLSR")
       rt_pkts + 1)
)
END (
print rt_pkts)
```

Gambar 4.23 *Pseudeucode* RO

Contoh perintah untuk analisis RO dari *trace file* model transmisi *TwoRayGround* dengan protokol OLSR dan kecepatan maksimal perpindahan *node* sebesar 5 m/s seperti Gambar 4.24.

```
awk -f ro.awk olsr_outtrga1.tr
```

Gambar 4.24 Perintah menjalankan skrip ro.awk

Contoh *output* dari menjalankan skrip di atas seperti pada Gambar 4.25.

```
total no of routing packets       7670
```

Gambar 4.25 Hasil *running* skrip ro.awk

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas mengenai pengujian dari skenario NS-2 yang telah dibuat. Pengujian fungsionalitas akan dibagi ke dalam beberapa skenario pengujian.

5.1. Lingkungan Pengujian

Uji coba dilakukan pada laptop yang telah terpasang dua buah sistem operasi yaitu Windows dan Linux. Spesifikasi komputer yang digunakan ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Komputer yang Digunakan

Komponen	**Spesifikasi**
CPU	Intel(R) Core(TM) i5-2450M CPU @2.50GHz
Sistem Operasi	Windows 7 Ultimate 64, Linux Ubuntu 10.04 LTS 64-bit (NS-2, OLSR, *Mobility Generation, Traffic-Connection Generation, TwoRayGround*, Nakagami)
Memori	4 GB DDR3
Media Penyimpanan	500 GB

5.2. Kriteria Pengujian

Pengujian pada skenario yang dihasilkan oleh *mobility generator default* dari NS-2 menggunakan beberapa kriteria. Pada Tabel 5.2 menunjukkan kriteria-kriteria yang ditentukan didalam skenario.

Tabel 5.2 Kriteria Pengujian

Kriteria	**Spesifikasi**
Skenario	MANET (*Random Way Point*), *mobility generator*
Kecepatan Maksimal Perpindahan *Node* (m/s)	5, 10, 15

Kriteria	**Spesifikasi**
Jumlah Percobaan	10 kali
Jarak *Node* 1 dan *Node* 2	Acak
Posisi Awal *Node*	Acak
Pergerakan	Acak
Protokol *Routing*	OLSR
Pengiriman Paket Data	*TwoRayGround*: 0 – 100 detik Nakagami : 100 – 200 detik

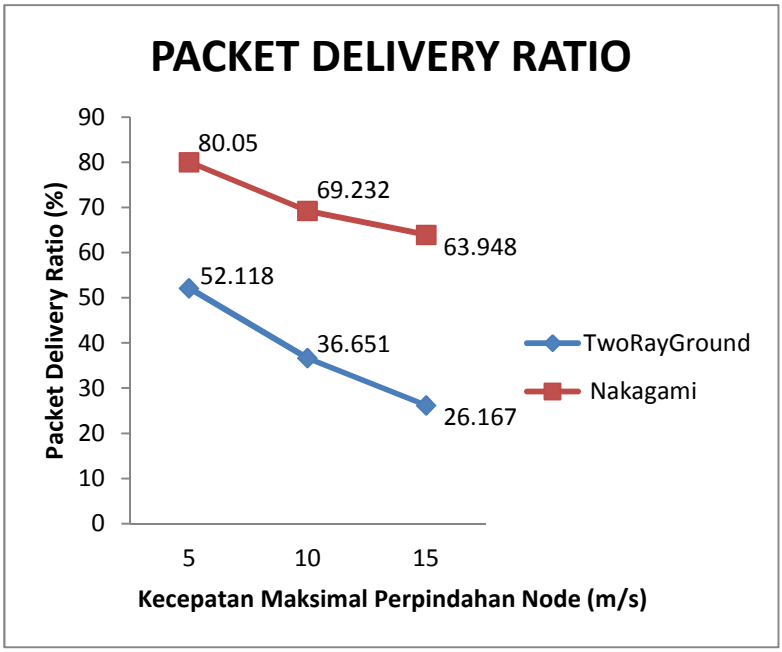
5.3. Analisis *Packet Delivery Ratio (PDR)*

Trace file hasil menjalankan program skenario *node-movement (mobility generation)* menggunakan model transmisi *TwoRayGround* dan Nakagami kemudian dianalisis nilai PDR melalui *script* pdr.awk. Hasil tiap perhitungan PDR skenario ditabulasikan dan dirata-ratakan menjadi seperti pada Tabel 5.3.

Tabel 5.3 PDR Skenario *Node-Movement*

Kecepatan Maksimal Perpindahan *Node* (m/s)	**PDR (%)**	
	TwoRayGround	**Nakagami**
5	52.118	80.050
10	36.651	69.232
15	26.167	63.948

Pada Tabel 5.3 dan Gambar 5.1 menunjukkan performa PDR yang dihasilkan oleh model transmisi *TwoRayGround* dan Nakagami pada jaringan MANET dengan menggunakan skenario *node-movement (mobility generaion)* yang bersifat *Random Way Point*. Terlihat bahwa PDR yang dihasilkan oleh kedua model transmisi semakin menurun dengan bertambahnya kecepatan maksimal perpindahan *node*.



Gambar 5.1 Grafik PDR Skenario *node-movement*

Nilai PDR yang dihasilkan oleh model transmisi Nakagami memiliki nilai yang lebih tinggi dari pada model transmisi *TwoRayGround* walaupun pada kedua model transmisi menghasilkan nilai PDR yang lebih baik pada saat kecepatan maksimal perpindahan *node* rendah daripada pada saat kecepatan maksimal perpindahan *node* tinggi. Namun pada saat kecepatan maksimal perpindahan *node* sebesar 5 m/s dan 10 m/s, model transmisi Nakagami hanya mengalami penurunan sebesar 10.818% sedangkan model transmisi *TwoRayGround* mengalami penurunan sebesar 15.467%. Begitu pula pada saat kecepatan maksimal perpindahan *node* sebesar 15 m/s, pada model transmisi Nakagami semakin menurun tetapi hanya sebesar 5.284% sedangkan pada model transmisi *TwoRayGround* sebesar 10.484%.

Terlihat bahwa nilai PDR yang dihasilkan oleh model transmisi Nakagami lebih baik dan lebih stabil daripada PDR yang dihasilkan oleh model transmisi *TwoRayGround* pada skenario *node-movement (mobility generator)* MANET. Nilai PDR yang hasilkan lebih stabil pada model transmisi Nakagami dikarenakan kemampuannya untuk mengidentifikasi/memperhatikan lingkungan sekitar area simulasi dan pengaruh ketinggian *antenna* pada *node* yang ada. Tidak seperti model transmisi *TwoRayGround*, dari skenario yang yang dihasilkan oleh *file node-movement (mobility generation)* tidak mampu untuk mengidentifikasi/memperhatikan lingkungan sekitar area simulasi dan pengaruh ketinggian *antenna* pada *node* yang menyebabkan pengiriman paket antar *node* yang lebih cepat dan dinamis. Pergerakan *node* yang lebih dinamis inilah yang memungkinkan terjadinya banyak rute putus saat pengiriman paket data ataupun kegagalan pembentukan tabel *routing* yang dibuat oleh OLSR sehingga menyebabkan paket data yang dikirimkan tidak sampai ke tujuan. Hal lain yang dapat mempengaruhi penurunan ataupun peningkatan nilai PDR yang dihasilkan adalah penempatan dan pergerakan secara acak yang di implementasikan pada skenario yang dihasilkan oleh *file node-movement (mobility generation)*.

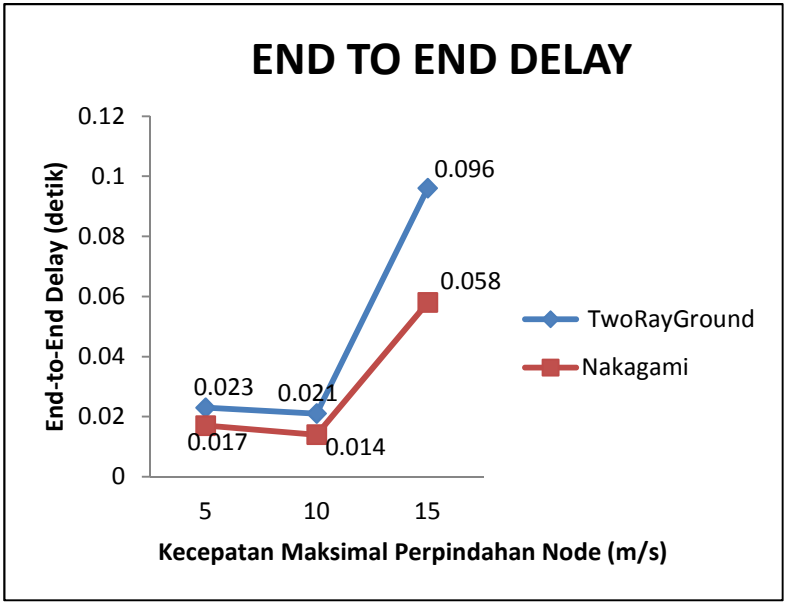
5.4. Analisis *End-to-End Delay (E2D)*

Trace file hasil menjalankan program skenario *node-movement (mobility generation)* menggunakan model transmisi *TwoRayGround* dan Nakagami kemudian dianalisis nilai *End-to-End Delay* melalui *script* endtoend.awk. Hasil tiap perhitungan E2D skenario ditabulasikan dan dirata-ratakan menjadi seperti pada Tabel 5.4.

Tabel 5.4 E2D Skenario *Node-Movement*

Kecepatan Maksimal Perpindahan *Node* (m/s)	**E2D (detik)**	
	TwoRayGround	**Nakagami**
5	0.023	0.017

Kecepatan Maksimal Perpindahan *Node* (m/s)	E2D (detik)	
	TwoRayGround	Nakagami
10	0.021	0.014
15	0.096	0.058



Gambar 5.2 Grafik E2D Skenario *node-movement*

Performa *End-to-End Delay* baik pada model transmisi *TwoRayGround* maupun Nakagami menunjukkan nilai yang fluktuatif dimana pada saat kecepatan maksimal perpindahan *node* sebesar 5 m/s, nilai E2D yang dihasilkan yaitu 0.023 dan 0.017 detik. Namun pada saat kecepatan maksimal perpindahan *node* sebesar 10 m/s, mengalami sedikit penurunan sebanyak 0.002 detik untuk model transmisi *TwoRayGround* dan 0.003 detik untuk model transmisi Nakagami kemudian naik secara tajam pada saat kecepatan maksimal perpindahan *node* sebesar 15

m/s, nilai E2D yang dihasilkan yaitu menjadi sebanyak 0.096 dan 0.058 detik.

Performa *End-to-End Delay* pada model transmisi Nakagami memiliki nilai yang lebih baik dan lebih stabil dibandingkan dengan performa E2D yang dihasilkan oleh model transmisi *TwoRayGround.* Hal dapat terjadi akibat mobilitas pengiriman paket antar *node* yang sangat dinamis pada skenario *node-movement (mobility generation)* yang dihasilkan oleh model transmisi *TwoRayGround.* Mobilitas yang sangat dinamis ini memungkinkan terjadinya rute putus saat pengiriman paket data sehingga pengiriman paket dimasukkan ke dalam antrian dan menunggu rute baru terbentuk sebelum pengiriman paket data dilanjutkan kembali.

5.5. Analisis *Routing Overhead (RO)*

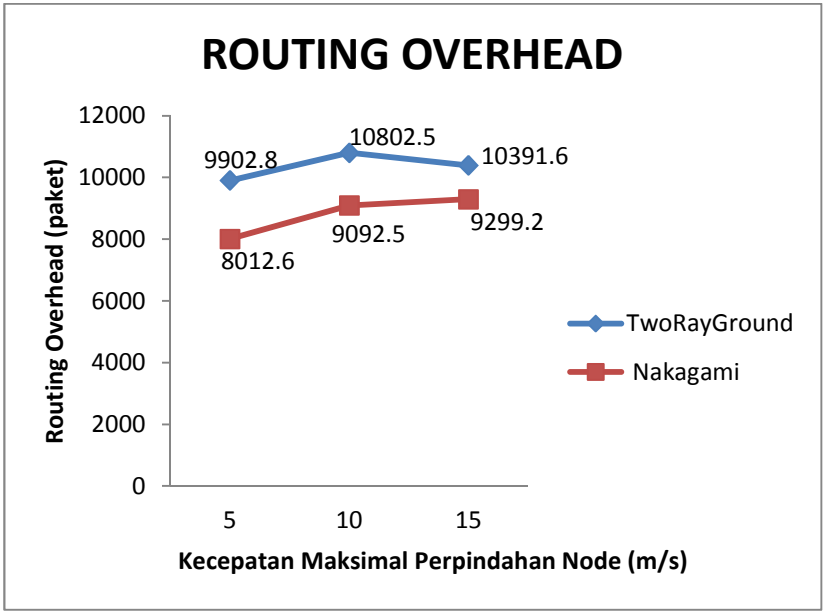
Trace file hasil menjalankan program skenario *node-movement (mobility generation)* menggunakan model transmisi *TwoRayGround* dan Nakagami kemudian dianalisis nilai *Routing Overhead* melalui *script* ro.awk. Hasil tiap perhitungan RO skenario ditabulasikan dan dirata-ratakan menjadi Tabel 5.5.

Tabel 5.5 RO Skenario *Node-Movement*

Kecepatan Maksimal Perpindahan *Node* (m/s)	***Routing* Overhead (Paket)**	
	TwoRayGround	**Nakagami**
5	9902.8	8012.6
10	10802.5	9092.5
15	10391.6	9299.2

Pada Tabel 5.5 dan Gambar 5.3 menunjukkan pengujian model transmisi *TwoRayGround* untuk nilai *Routing Overhead* yang dihasilkan memiliki nilai yang fluktuatif berdasarkan penambahan kecepatan maksimal perpindahan *node* yang terdapat dalam simulasi. Pada saat kecepatan maksimal perpindahan *node* sebesar 10 m/s terjadi peningkatan nilai *Routing Overhead* yaitu sebanyak 899.7 paket. Namun, kemudian menurun pada saat kecepatan maksimal perpindahan *node* sebesar 15 m/s yaitu

sebanyak 410.9 paket. Sedangkan untuk model transmisi Nakagami mengalami peningkatan. Pada saat kecepatan maksimal perpindahan *node* sebesar 10 m/s dan 15 m/s terjadi peningkatan nilai *Routing Overhead* yaitu sebanyak 1079.9 dan 206.7 paket.



Gambar 5.3 Grafik RO Skenario *node-movement*

Nilai *Routing Overhead* yang dihasilkan oleh model transmisi *TwoRayGround* memiliki nilai yang lebih besar dibandingkan dengan nilai *Routing Overhead* yang dihasilkan oleh model transmisi Nakagami. Hal ini terjadi karena pengiriman paket *routing* berjenis *send* maupun *forward* yang dihasilkan oleh *TwoRayGround* lebih banyak daripada Nakagami. Dengan demikian semakin banyaknya paket *routing send* dan *forward* yang dihasilkan, pengiriman paket data yang dilakukan memiliki kemungkinan yang lebih besar untuk sampai ke *node* tujuan.

Hasil analisis yang dilakukan pada metrik-metrik diatas menyebabkan perbedaan diantara model transmisi yang digunakan, yaitu *TwoRayGround* dan Nakagami. Beberapa faktor yang menyebabkan terjadinya perbedaan ini mengacu pada lokasi dan pergerakan pada *node*.

LAMPIRAN

```
#
# nodes: 50, pause: 10.00, max speed: 5.00,
max x: 510.00, max y: 510.00
#
$node_(0) set X_ 448.902173976333
$node_(0) set Y_ 416.404343429511
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 473.407075320993
$node_(1) set Y_ 226.655608269937
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 311.019492566598
$node_(2) set Y_ 242.125035054518
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 182.819681828025
$node_(3) set Y_ 47.084381788399
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 473.631282775967
$node_(4) set Y_ 493.020276615447
$node_(4) set Z_ 0.000000000000
$node_(5) set X_ 72.128431359938
$node_(5) set Y_ 98.745046693126
$node_(5) set Z_ 0.000000000000
$node_(6) set X_ 219.848949226686
$node_(6) set Y_ 500.880808684325
$node_(6) set Z_ 0.000000000000
$node_(7) set X_ 488.664168895257
$node_(7) set Y_ 354.386564756838
$node_(7) set Z_ 0.000000000000
$node_(8) set X_ 462.248912960704
$node_(8) set Y_ 359.831124242599
$node_(8) set Z_ 0.000000000000
$node_(9) set X_ 111.594985467791
$node_(9) set Y_ 168.102204449302
$node_(9) set Z_ 0.000000000000
$node_(10) set X_ 138.773537593637
$node_(10) set Y_ 116.677898866824
$node_(10) set Z_ 0.000000000000
$node_(11) set X_ 469.802484423601
$node_(11) set Y_ 425.207551641662
$node_(11) set Z_ 0.000000000000
$node_(12) set X_ 440.669712028303
$node_(12) set Y_ 30.701673965889
```

```
43  $node_(12) set Z_ 0.000000000000
44  $node_(13) set X_ 88.516065262105
45  $node_(13) set Y_ 182.470727535875
46  $node_(13) set Z_ 0.000000000000
47  $node_(14) set X_ 473.346147249918
48  $node_(14) set Y_ 434.697213120996
49  $node_(14) set Z_ 0.000000000000
50  $node_(15) set X_ 307.130970678581
51  $node_(15) set Y_ 154.193753810204
52  $node_(15) set Z_ 0.000000000000
53  $node_(16) set X_ 170.270852210544
54  $node_(16) set Y_ 303.584369659784
55  $node_(16) set Z_ 0.000000000000
56  $node_(17) set X_ 95.507005400829
57  $node_(17) set Y_ 291.955371572803
58  $node_(17) set Z_ 0.000000000000
59  $node_(18) set X_ 428.620863050022
60  $node_(18) set Y_ 385.637507113302
61  $node_(18) set Z_ 0.000000000000
62  $node_(19) set X_ 427.228938416489
63  $node_(19) set Y_ 19.934254157433
64  $node_(19) set Z_ 0.000000000000
65  $node_(20) set X_ 135.431084548977
66  $node_(20) set Y_ 17.883010720756
67  $node_(20) set Z_ 0.000000000000
```

Gambar 7.1 Posisi *node* dari potongan Skenario

```
1   $god_ set-dist 0 1 1
2   $god_ set-dist 0 2 1
3   $god_ set-dist 0 3 2
4   $god_ set-dist 0 4 1
5   $god_ set-dist 0 5 3
6   $god_ set-dist 0 6 1
7   $god_ set-dist 0 7 1
8   $god_ set-dist 0 8 1
9   $god_ set-dist 0 9 2
11  $god_ set-dist 0 10 2
12  $god_ set-dist 0 11 1
13  $god_ set-dist 0 12 2
14  $god_ set-dist 0 13 2
15  $god_ set-dist 0 14 1
16  $god_ set-dist 0 15 2
17  $god_ set-dist 0 16 2
```

```
18  $god_ set-dist 0 17 2
19  $god_ set-dist 0 18 1
20  $god_ set-dist 0 19 2
21  $god_ set-dist 0 20 3
22  $god_ set-dist 0 21 1
23  $god_ set-dist 0 22 2
24  $god_ set-dist 0 23 2
25  $god_ set-dist 0 24 1
26  $god_ set-dist 0 25 1
27  $god_ set-dist 0 26 2
28  $god_ set-dist 0 27 2
29  $god_ set-dist 0 28 2
30  $god_ set-dist 0 29 2
31  $god_ set-dist 0 30 2
32  $god_ set-dist 0 31 2
33  $god_ set-dist 0 32 2
34  $god_ set-dist 0 33 2
35  $god_ set-dist 0 34 2
36  $god_ set-dist 0 35 3
37  $god_ set-dist 0 36 1
38  $god_ set-dist 0 37 1
39  $god_ set-dist 0 38 2
40  $god_ set-dist 0 39 2
41  $god_ set-dist 0 40 3
42  $god_ set-dist 0 41 1
43  $god_ set-dist 0 42 2
44  $god_ set-dist 0 43 2
45  $god_ set-dist 0 44 3
46  $god_ set-dist 0 45 2
47  $god_ set-dist 0 46 2
48  $god_ set-dist 0 47 2
49  $god_ set-dist 0 48 2
50  $god_ set-dist 0 49 1
51  $god_ set-dist 1 2 1
52  $god_ set-dist 1 3 2
53  $god_ set-dist 1 4 2
54  $god_ set-dist 1 5 2
55  $god_ set-dist 1 6 2
56  $god_ set-dist 1 7 1
57  $god_ set-dist 1 8 1
58  $god_ set-dist 1 9 2
59  $god_ set-dist 1 10 2
60  $god_ set-dist 1 11 1
61  $god_ set-dist 1 12 1
```

```
62  $god_ set-dist 1 13 2
63  $god_ set-dist 1 14 1
64  $god_ set-dist 1 15 1
65  $god_ set-dist 1 16 2
66  $god_ set-dist 1 17 2
67  $god_ set-dist 1 18 1
68  $god_ set-dist 1 19 1
69  $god_ set-dist 1 20 2
70  $god_ set-dist 1 21 1
71  $god_ set-dist 1 22 2
72  $god_ set-dist 1 23 1
73  $god_ set-dist 1 24 1
74  $god_ set-dist 1 25 1
75  $god_ set-dist 1 26 2
76  $god_ set-dist 1 27 2
77  $god_ set-dist 1 28 1
78  $god_ set-dist 1 29 1
79  $god_ set-dist 1 30 1
80  $god_ set-dist 1 31 2
81  $god_ set-dist 1 32 1
82  $god_ set-dist 1 33 2
83  $god_ set-dist 1 34 2
84  $god_ set-dist 1 35 3
85  $god_ set-dist 1 36 1
86  $god_ set-dist 1 37 2
87  $god_ set-dist 1 38 1
88  $god_ set-dist 1 39 2
89  $god_ set-dist 1 40 2
90  $god_ set-dist 1 41 1
91  $god_ set-dist 1 42 3
92  $god_ set-dist 1 43 2
93  $god_ set-dist 1 44 2
94  $god_ set-dist 1 45 3
95  $god_ set-dist 1 46 1
96  $god_ set-dist 1 47 2
97  $god_ set-dist 1 48 2
98  $god_ set-dist 1 49 2
```

Gambar 7.2 Pembuatan GOD setiap *node* dari potongan Skenario

```
1   $ns_ at 10.000000000000 "$node_(0) setdest
    191.734433653942 125.212223281651
    0.345476796352"
2   $ns_ at 10.000000000000 "$node_(1) setdest
    125.972955560399 280.372260063288
    3.433236813665"
3   $ns_ at 10.000000000000 "$node_(2) setdest
    304.565395484586 296.333151906099
    2.639912491222"
4   $ns_ at 10.000000000000 "$node_(3) setdest
    187.645145254254 412.497638447561
    2.110040152652"
5   $ns_ at 10.000000000000 "$node_(4) setdest
    26.512936466534 377.731896042413
    0.287293249219"
6   $ns_ at 10.000000000000 "$node_(5) setdest
    91.692611126244 484.529824648566
    0.809873247858"
7   $ns_ at 10.000000000000 "$node_(6) setdest
    3.255450890198 210.700281822509
    0.443639752287"
8   $ns_ at 10.000000000000 "$node_(7) setdest
    4.462151485059 443.587156789407
    1.009461341884"
9   $ns_ at 10.000000000000 "$node_(8) setdest
    136.167459183566 328.435913833014
    1.349484941618"
10  $ns_ at 10.000000000000 "$node_(9) setdest
    444.906867719402 309.072393471621
    2.503900049676"
11  $ns_ at 10.000000000000 "$node_(10) setdest
    216.709987622611 414.705440853427
    1.011016815969"
12  $ns_ at 10.000000000000 "$node_(11) setdest
    145.178860134260 239.110360987241
    3.278821148443"
13  $ns_ at 10.000000000000 "$node_(12) setdest
    43.800259416563 411.536704401866
    1.294917889019"
14  $ns_ at 10.000000000000 "$node_(13) setdest
    34.275357795245 75.557407547240
    3.818000060074"
15  $ns_ at 10.000000000000 "$node_(14) setdest
    418.099233895689 229.795209308482
```

```
    4.812053536509"
16  $ns_ at 10.000000000000 "$node_(15) setdest
    509.636967106371 131.481084925604
    2.063373914962"
```

Gambar 7.3 Pergerakan setiap *node* dari potongan Skenario

```
1   $ns_ at 10.180742080271 "$god_ set-dist 6 20
    2"
2   $ns_ at 10.180742080271 "$god_ set-dist 20 34
    1"
3   $ns_ at 10.302439202349 "$god_ set-dist 2 14
    1"
4   $ns_ at 10.302439202349 "$god_ set-dist 3 14
    2"
5   $ns_ at 10.302439202349 "$god_ set-dist 14 39
    2"
6   $ns_ at 10.317997015819 "$god_ set-dist 4 27
    2"
7   $ns_ at 10.317997015819 "$god_ set-dist 27 37
    1"
8   $ns_ at 10.550030215660 "$god_ set-dist 13 24
    2"
9   $ns_ at 10.816437107569 "$god_ set-dist 11 21
    1"
10  $ns_ at 10.816437107569 "$god_ set-dist 11 31
    2"
11  $ns_ at 11.178419469665 "$god_ set-dist 11 25
    1"
12  $ns_ at 11.439036720008 "$god_ set-dist 1 35
    2"
13  $ns_ at 11.439036720008 "$god_ set-dist 28 35
    2"
14  $ns_ at 11.439036720008 "$god_ set-dist 30 35
    2"
15  $ns_ at 11.439036720008 "$god_ set-dist 35 46
    1"
16  $ns_ at 11.545889022671 "$god_ set-dist 17 22
    2"
17  $ns_ at 11.727051439230 "$god_ set-dist 2 12
    2"
18  $ns_ at 11.727051439230 "$god_ set-dist 12 17
    3"
19  $ns_ at 11.911846168064 "$god_ set-dist 16 46
```

```
    1"
20  $ns_ at 11.911846168064 "$god_ set-dist 42 46
    2"
```

Gambar 7.4 iInformasi pada GOD dari potongan Skenario

```
1   #
2   # nodes: 10, max conn: 8, send rate: 0.25,
3   seed: 1.0
4   #
5   #
6   # 1 connecting to 2 at time 2.5568388786897245
7   #
8   set udp_(0) [new Agent/UDP]
9   $ns_ attach-agent $node_(1) $udp_(0)
10  set null_(0) [new Agent/Null]
11  $ns_ attach-agent $node_(2) $null_(0)
12  set cbr_(0) [new Application/Traffic/CBR]
13  $cbr_(0) set packetSize_ 512
14  $cbr_(0) set interval_ 1
15  $cbr_(0) set random_ 1
16  $cbr_(0) set maxpkts_ 10000
17  $cbr_(0) attach-agent $udp_(0)
18  $ns_ connect $udp_(0) $null_(0)
19  $ns_ at 2.5568388786897245 "$cbr_(0) start"
20  #
```

Gambar 7.5 Koneksi yang digunakan pada cbrtest.txt

```
1   ========================
2   # Define options
3   # ======================
4   set val(chan)  Channel/WirelessChannel
5   set val(prop)  Propagation/TwoRayGround
6   set val(netif) Phy/WirelessPhy
7   set val(mac)   Mac/802_11
8   set val(ifq)   Queue/DropTail/PriQueue
9   set val(ll)    LL
10  set val(ant)   Antenna/OmniAntenna
11  set opt(x)     510  ;# X dimension of the
    topography
12  set opt(y)     510  ;# Y dimension of the
    topography
13  set val(ifqlen) 50  ;# max packet in ifq
```

```
13  set val(nn)  50 ;# how many nodes are
    simulated
14  set val(seed) 0.0
15  set val(adhocRouting) OLSR
16  set val(stop)         100 ;# simulation time
17  set val(cp)  "cbrtest.txt" ;#<-- traffic file
18  set val(sc)  "scena1.txt"  ;#<-- mobility file
19
20  Phy/WirelessPhy     set    RXThresh_ 1.42681e-
    08 ;#100m
21  #
22  ================================================
    ========================
23  # Main Program
24  #
25  ================================================
    ========================
26  # Initialize Global Variables
27  # create simulator instance
28
29  set ns_      [new Simulator]
30
31  # create trace object for ns and nam
32
33  set tracefd    [open olsr_outtrga10.tr w]
34  set namtrace   [open olsr_outtrga10.nam w]
35
36  $ns_ trace-all $tracefd
37  $ns_ namtrace-all-wireless $namtrace $opt(x)
38  $opt(y)
39
40  # set up topology object
41  set topo            [new Topography]
42  $topo load_flatgrid $opt(x) $opt(y)
43
44  # Create God
45  set god_ [create-god $val(nn)]
46
47  #global node setting
48  $ns_ node-config -adhocRouting
49  $val(adhocRouting) \
50                   -llType $val(ll) \
51                   -macType $val(mac) \
52                   -ifqType $val(ifq) \
```

```
53                  -ifqLen $val(ifqlen) \
54                  -antType $val(ant) \
55                  -propType $val(prop) \
56                  -phyType $val(netif) \
57                  -channelType $val(chan) \
58                  -topoInstance $topo \
59                  -agentTrace ON \
60                  -routerTrace ON \
61                  -macTrace OFF \
62                  -movementTrace ON \
63
64  #  Create the specified number of nodes
65  [$val(nn)] and "attach" them
66  #  to the channel.
67  for {set i 0} {$i < $val(nn)} {incr i} {
68              set node_($i) [$ns_ node]
69              $node_($i) random-motion 0 ;#
    disable random motion
70  }
71
72  # Define node movement model
73  puts "Loading connection pattern..."
74  source $val(cp)
75
76  # Define traffic model
77  puts "Loading scenario file..."
78  source $val(sc)
79
80  # Define node initial position in nam
81
82  for {set i 0} {$i < $val(nn)} {incr i} {
83
84      # 20 defines the node size in nam, must
    adjust it according to your scenario
85      # The function must be called after
    mobility model is defined
86
87      $ns_ initial_node_pos $node_($i) 20
88  }
89
90  # Tell nodes when the simulation ends
91  for {set i 0} {$i < $val(nn) } {incr i} {
92      $ns_ at $val(stop).0 "$node_($i) reset";
93  }
```

```
93
94  #$ns_ at  $val(stop)       "stop"
95  $ns_ at  $val(stop).0002 "puts \"NS
    EXITING...\" ; $ns_ halt"
96
97  puts $tracefd "M 0.0 nn $val(nn) x $opt(x) y
    $opt(y) rp $val(adhocRouting)"
98  puts $tracefd "M 0.0 sc $val(sc) cp $val(cp)
    seed $val(seed)"
99  puts $tracefd "M 0.0 prop $val(prop) ant
    $val(ant)"
100
101 puts "Starting Simulation..."
102 $ns_ run
```

Gambar 7.6 *File* .tcl untuk Protokol *Routing* OLSR

```
1   BEGIN {
2   sent=0;
3   recv=0;
4   pdr=0;
5   }
6   {
7   #count packet send
8   if ($1 == "s" && $3 == "_1_" && $4 == "AGT" &&
    $7 == "cbr")
9         {
10        sent++;
11        }
12  #count packet receive
13  if ($1 == "r" && $3 == "_2_" && $4 == "AGT" &&
    $7 == "cbr")
14        {
15        recv++;
16        }
17  }
18  END {
19  pdr = ( recv / sent ) * 100
20  print "Transmitted packet (s):", sent;
21  print "Received packet (s):", recv;
22  print "Packet delivery ratio:", pdr, "%";
23  }
```

Gambar 7.7 Implementasi *Packet Delivery Ratio*

```
1   BEGIN {
2   rt_pkts = 0;
3   }
4   {
5   if (($1 == "s" || $1 == "f") && ($4 == "RTR")
    && ($7 == "OLSR"))
6       rt_pkts++;
7   }
8   END {
9   printf     ("Total     number     of     routing
10  packets\t%d\n", rt_pkts);
11  }
```

Gambar 7.8 Implementasi *Routing Overhead*

```
1   BEGIN{
2   for ( i in pkt_id)
3        {
4                pkt_id[i] = 0;
5             }
6   for ( i in pkt_sent)
7        {
8                pkt_sent[i] = 0;
9             }
10  for ( i in pkt_recv )
11       {
12               pkt_recv[i] = 0;
13            }
14       delay = avg_delay = 0;
15       recv = 0;
16       recv_id = 0;
17  }
18  {
19  # count packet send
20  if ( $1 == "s" && $3 == "_1_" && $4 == "AGT"
    && $7 == "cbr" )
21  {
22    pkt_sent[$6] = $2;
23  }
24  # count packet receive
25  if ( $1 == "r" && $3 == "_2_" && $4 == "AGT"
    && $7 == "cbr" &&  recv_id != $6 )
26  {
```

```
27    recv++;
28    recv_id = $6;
29    pkt_recv[$6] = $2;
30  }
31  }
32  END{
33  for (i in pkt_recv)
34      {
35        delay += pkt_recv[i] - pkt_sent[i];
36      }
37
38  avg_delay = delay / recv;
39
40  print "Total Packet(s) Receive =", recv;
41  print "Total Delay =", delay, "second";
42  print  "Average  Packet  Delivery  Delay  =  ",
43  avg_delay, "second";
44  }
```

Gambar 7.9 Implementasi *End-to-End Delay*

Instalasi NS-2

Dokumentasi tentang cara instalasi NS-2 dan proses *patching routing* protokol OLSR bersumber dari nsnam.com. Pada Tugas Akhir ini digunakan cara mengunduh *file* NS-2 dan *patch* OLSR yang disediakan oleh nsnam.com. Langkah-langkah detail adalah sebagai berikut.

- Pertama kali lakukan *update* komponen terbaru dari Ubuntu dan pastikan Ubuntu yang diinstal ter-*update* seluruhnya.

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get update
```

Gambar 7.10 Perintah *update* seluruh komponen Ubuntu

- Kemudian lakukan instalasi modul-modul dependensi dari NS-2 yaitu build-essential, autoconf, automake, tcl8.5-dev tk8.5-dev, perl, xgraph, libxt-dev, libx11-dev, libxmu-dev dan gcc-4.4.

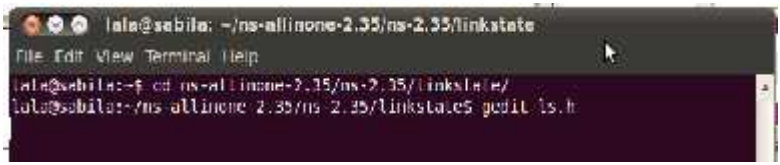
```
$ sudo apt-get install build-essential autoconf
automake
$ sudo apt-get install tcl8.5-dev tk8.5-dev
$ sudo apt-get install perl xgraph libxt-dev
libx11-dev libxmu-dev
$ sudo apt-get install gcc-4.4
```

Gambar 7.11 Perintah instalasi dependensi NS-2

- Setelah semua dependensi lengkap, *file* ns-2 yang telah diunduh dipindahkan menuju *folder* /home dan dilakukan proses ektraksi. Kemudian dilakukan proses pengubahan script pada ls.h yang terdapat pada *folder* /ns-allinone-2.35/ns-2.35/linkstate/ls.h.

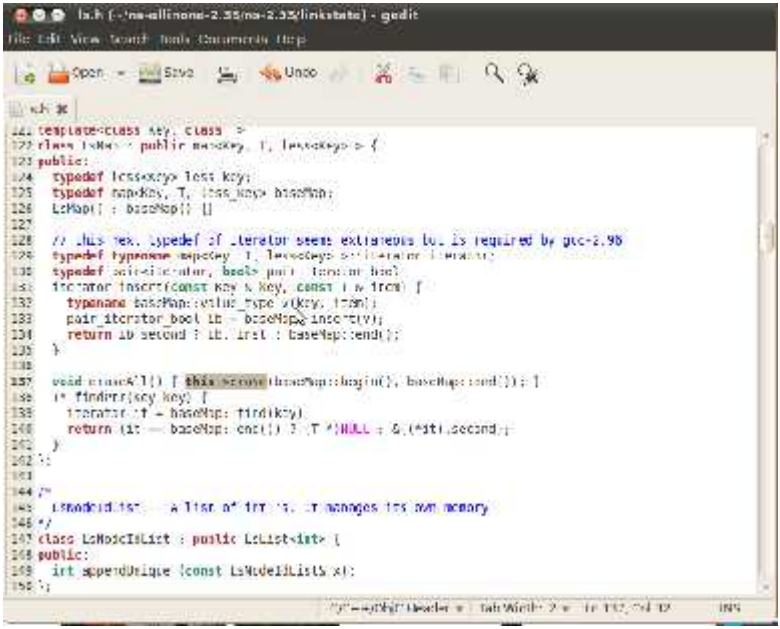
```
$ tar -xvzf ns-allinone-2.35.tar.gz
$ cd /ns-allinone-2.35/ns-2.35/linkstate/
$ gedit ls.h
```

Gambar 7.12 Proses ekstrak dan pengubahan ls.h



Gambar 7.13 Screenshot proses ekstrak dan pengubahan ls.h

- Setelah semua dependensi lengkap, *file* ns-2 yang telah diunduh dipindahkan menuju *folder* /home dan dilakukan proses ektraksi. Kemudian dilakukan proses pengubahan Pada *line* ke-137, *erase* diubah menjadi *this->erase* karena jika tidak maka akan terjadi kegagalan pada saat instalasi NS-2.



Gambar 7.14 Proses pengubahan *line of code* ls.h

- Kemudian dilakukan patching OLSR pada NS-2 dengan cara memasukkan patch yang telah di unduh ke dalam *folder* /ns-allinone-2.35/ns-2.35/ kemudian dilakukan ekstraksi dan proses *patching*.

```
$ cd ns-allinone-2.35/ns-2.35/
$ tar zxvf um-olsr-1.0.tgz
$ ln -s ./um-olsr-1.0 ./olsr
$ patch -p1 < olsr/um-olsr_ns-2.35_v1.0.patch
```

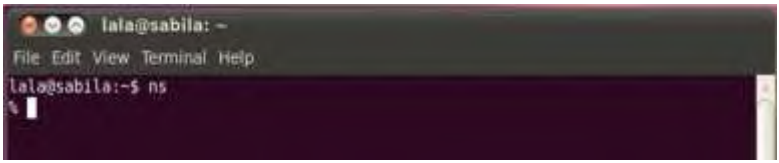
Gambar 7.15 Perintah *patching* OLSR

- Terakhir dilakukan proses instalasi dari NS-2 yang telah di *patching* dengan OLSR.

```
$ sudo ./install
```

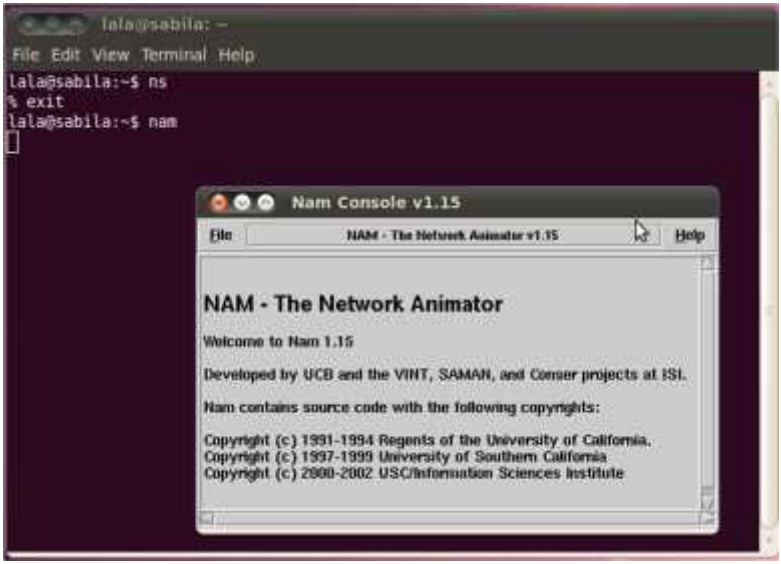
Gambar 7.16 Perintah instalasi NS-2

- Untuk melakukan tes apakah NS-2 telah terinstall dengan baik maka dapat dilakukan dengan mengetikkan command ‘ns’ pada terminal dan apabila muncul tanda ‘%’ pada terminal berarti NS-2 telah terinstall dengan baik.



Gambar 7.17 Perintah pengecekan NS-2

- Untuk melakukan tes apakah NAM telah terinstall dengan baik maka dapat dilakukan dengan mengetikkan command ‘nam’ pada terminal seperti pada Gambar 7.18.



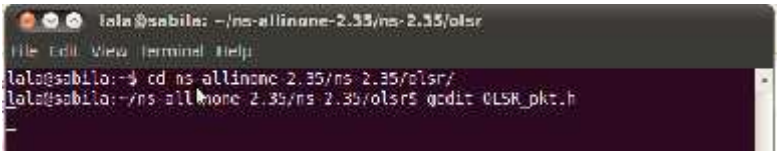
Gambar 7.18 Perintah pengecekan NAM

- Kemudian apabila ingin melakukan percobaan dengan jumlah *node* sebanyak 75 atau 100, terlebih dahulu kita ubah *file*

OLSR_pkt.h pada *pacth* OLSR yang telah terinstal sebelumnya.

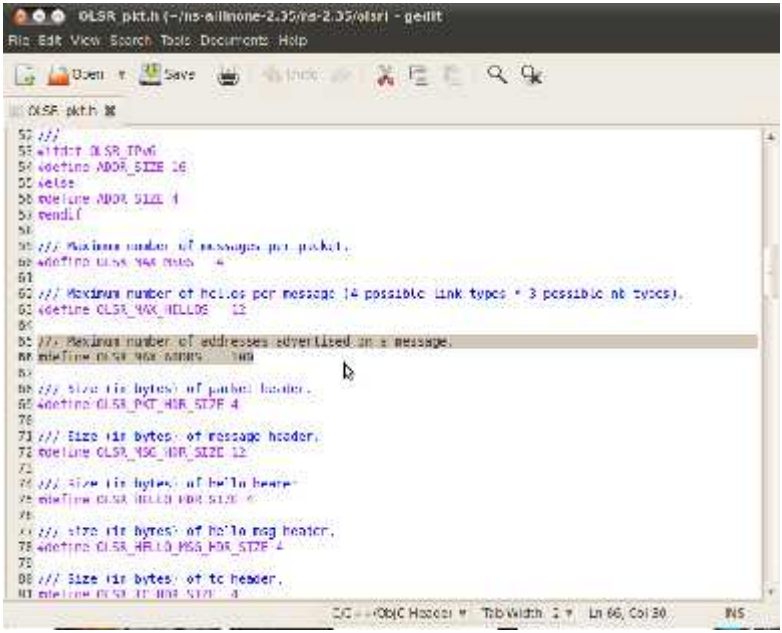
```
$ cd ns-allinone-2.35/ns-2.35/olsr/
$ gedit OLSR_pkt.h
```

Gambar 7.19 Perintah untuk pengubahan OLSR_pkt.h



Gambar 7.20 Screenshot pengubahan OLSR_pkt.h

- Pada line ke-66, 64 diubah menjadi 100 karena jika tidak akan terjadi kegagalan "*Simulation Error: Segmentation Fault*" pada saat *run file* .tcl dari OLSR apabila melakukan percobaan dengan jumlah *node* sebanyak 75 atau 100.



Gambar 7.21 Proses pengubahan *line of code* OLSR_pkt.h

[Halaman ini sengaja dikosongkan]

BAB VI
PENUTUP

Pada bab ini diberikan kesimpulan yang diambil selama pengerjaan Tugas Akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1. Kesimpulan

Kesimpulan yang dapat diambil dalam Tugas Akhir ini adalah sebagai berikut:

1. Skenario MANET yang dihasilkan oleh *node-movement (mobility generation)* dan dijalankan menggunakan model transmisi *TwoRayGround* dengan penambahan kecepatan maksimal perpindahan *node* memiliki performa sebagai berikut:
   - Performa *Packet Delivery Ratio* yang dihasilkan semakin menurun secara stabil pada rentang nilai 52.118% menjadin 26.167% dari kecepatan maksimal perpindahan *node* sebesar 5 m/s hingga 15 m/s.
   - Performa *End to End Delay* yang dihasilkan memiliki nilai *delay* fluktuatif , yaitu sedikit menurun pada saat kecepatan maksimal perpindahan *node* sebesar 10 m/s namun naik secara tajam pada saat kecepatan maksimal perpindahan *node* sebesar 15 m/s.
   - *Routing Overhead* yang dihasilkan memiliki nilai yang tidak stabil, yaitu terjadi kenaikan pada saat kecepatan maksimal perpindahan *node* sebesar 5 m/s namun menurun pada saat kecepatan maksimal perpindahan *node* sebesar 15 m/s.
2. Skenario MANET yang dihasilkan oleh *node-movement (mobility generation)* dan dijalankan menggunakan model transmisi Nakagami dengan penambahan kecepatan maksimal perpindahan *node* memiliki performa sebagai berikut:

- Performa *Packet Delivery Ratio* yang dihasilkan semakin menurun secara stabil pada rentang nilai 80.05% menjadi 63.948% dari kecepatan maksimal perpindahan *node* sebesar 5 m/s hingga 15 m/s.
- Performa *End to End Delay* yang dihasilkan memiliki nilai *delay* fluktuatif , yaitu sedikit menurun pada saat kecepatan maksimal perpindahan *node* sebesar 10 m/s namun naik secara tajam pada saat kecepatan maksimal perpindahan *node* sebesar 15 m/s.
- *Routing Overhead* yang dihasilkan semakin bertambah secara stabil, yaitu pada rentang nilai 8012.6 paket menjadi 9299.2 paket dari kecepatan maksimal perpindahan *node* sebesar 5 m/s hingga 15 m/s.

3. Hal – hal yang dapat mempengaruhi nilai PDR, E2D dan RO yang dihasilkan dari model transmisi *TwoRayGround* dan Nakagami adalah:
   - Posisi awal *node* yang dibuat secara acak.
   - Pergerakan *node* yang dibuat secara acak.
   - Lingkungan jaringan yang digunakan.
   - Keadaan sekitar simulasi.
   - Ketinggian antena
4. Dari hasil percobaan pada skenario MANET, didapatkan nilai PDR, E2D dan RO dengan model transmisi Nakagami lebih baik dan bagus daripada nilai PDR, E2D dan RO dengan model transmisi *TwoRayGround* sehingga:
   - Model transmisi *TwoRayGround* yang tidak memperhatikan keadaan sekitar simulasi dan ketinggian antena sehingga cocok untuk lingkungan MANET yang bentuk jaringannya lebih besar dan bersifat global.
   - Model transmisi Nakagami yang memperhatikan keadaan sekitar simulasi seperti gedung, pohon, tanah dan *obstacle-obstacle* lainnya serta ketinggian antena sehingga cocok untuk lingkungan VANET karena merepresentasikan bentuk jaringannya yang lebih detail daripada MANET.

6.2. Saran

Dalam pengerjaan Tugas Akhir ini terdapat beberapa saran untuk perbaikan serta pengembangan sistem yang telah dikerjakan sebagai berikut:

1. Dapat dilakukan percobaan pada lingkungan VANET untuk penerapan kedua model transmisi yaitu *TwoRayGround* dan Nakagami.
2. Dapat dilakukan pengurangan atau penambahan jumlah *node* dan penambahan jumlah percobaan untuk skenario *node-movement (mobility generation).*
3. Dapat dilakukan modifikasi pada parameter-parameter yang digunakan untuk membangkitkan uji coba MANET pada NS-2 seperti modifikasi pada parameter *transmission range*, modifikasi pada *pause time* selama waktu simulasi berlangsung.
4. Dapat dilakukan modifikasi protokol OLSR agar bersifat adaptif terhadap perubahan mobilitas *node* yang tinggi.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] I. N. B. Hartawan, “Optimasi Pemilihan Multi-Point Relay dengan Congestion Detection dalam Optimized Link State Routing pada Mobile Ad-Hoc Network,” *Digilib ITS*, p. 1, 2014.

[2] M. Affandes, "3. Mobile Ad Hoc Network (MANET)," 30 October 2011. [Online]. Available: https://affandezone.wordpress.com/2011/10/30/3-routing-pada-manet/. [Accessed 26 May 2016].

[3] A. Suprapto, "Mengenal Mobile Ad Hoc Network (MANET)," 7 September 2014. [Online]. Available: http://agungsuprapto.net/mengenal-mobile-ad-hoc-network-manet/. [Accessed 26 May 2016].

[4] S. C. Chan, "Jenis Jaringan Ad Hoc," 11 October 2014. [Online]. Available: http://www.materi-it.com/2014/11/jenis-jaringan-ad-hoc.html. [Accessed 26 May 2016].

[5] R. W., "Mobile Ad hoc Network (MANET)," 17 October 2012. [Online]. Available: http://ramdit.blogspot.co.id/2012/10/mobile-ad-hoc-network-manet.html. [Accessed 26 May 2016].

[6] Rechnernetze und Telematik University of Freiburg, 2011. [Online]. Available: http://archive.cone.informatik.uni-freiburg.de/teaching/vorlesung/manet-s07/exercises/DSDV.ppt. [Accessed 26 May 2016].

[7] A. Saputra, "Pengetahuan Tentang Jaringan," 25 March 2009. [Online]. Available: http://de-monk.blogspot.co.id/. [Accessed 29 March 2016].

[8] Pajala, Elina, Isotalo, Tero and dkk, "An improved simulation model for Nakagami-m-m fading channels for satellite positioning applications," Finlandia, Institute of Communications Engineering Tampere University of

Technology, p. 82.

[9] R. Baumann, Engineering and simulation of mobile ad hoc routing protocols for VANET on Highways and in cities, Institute of Technology Zurich, 2014.

[10] D. A. Maltz, "Simulation and Implementation," in *On-Demand Routing in Multi-Hop Wireless Mobile Ad Hoc Network*, School of Computer Science Carnegie Mellon University, 2001.

[11] 8 November 2014. [Online]. Available: http://bengkelubuntu.org/teks/awk/Praktikum%2001%20-%20Berkenalan%20dengan%20AWK.pdf.. [Accessed 29 May 2016].

BIODATA PENULIS

Dhiya'an Sabila Ramadhani, biasa dipanggil Lala, lahir di Bandung pada 28 Maret 1992. Penulis adalah anak pertama dari empat bersaudara dan dibesarkan di Madiun, Jawa Timur. Penulis menempuh pendidikan formal di MI Islamiyah 02 Madiun (1998-2004), SMPN 2 Madiun (2004-2007), SMAN 2 Madiun (2007-2010). Pada tahun 2010, penulis memulai pendidikan S1 Jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember Surabaya angkatan 2010 yang terdaftar dengan NRP 5110100017.

Di jurusan Teknik Informatika, penulis mengambil bidang minat Arsitektur Jaringan Komputer (AJK) dan memiliki ketertarikan di bidang *Operating System Linux*, Jaringan Komputer, Sistem Terdistribusi, Jaringan Multimedia, dan Teknik Kompresi. Selama menempuh kuliah, penulis juga aktif dalam organisasi kemahasiswaan seperti Himpunan Mahasiswa Teknik Computer (HMTC). Penulis dapat dihubungi melalui alamat email sabilalala92@gmail.com.